LOGISTICS OF CLINICAL TESTING

by

Eda Yücel

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Industrial Engineering and Operations Management

Koç University

December, 2011

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Eda Yücel

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Asst. Prof. Sibel Salman (Thesis Supervisor)

Assoc. Prof. Necati Aras

Assoc. Prof. Metin Türkay

Assoc. Prof. Lerzan Örmeci

Assoc. Prof. Alper Yıldırım

Date:

To my sons Mert Selim and Mete Kerem

ABSTRACT

In the delivery of healthcare services, important diagnostic and therapeutic decisions are often based on the results of clinical tests. Clinical testing laboratories aim to achieve timely and error free processing of the tests by means of coordinated collection, transportation, and processing of the specimens obtained from the patients. This thesis addresses the specimen collection problem that clinical testing laboratories face daily. In this problem, specimens that accumulate throughout the working day at customer sites should be transported to a facility for subsequent processing on equipment with limited capacity. We formalize the resulting routing and scheduling problem as collection for processing problem (CfPP). We analyze several fundamental properties of the problem, develop a linear mixed integer programming (MIP) model, and propose a number of heuristic approaches to identify effective collection strategies.

For the multi-vehicle problem with the makespan objective, we develop a polynomial-time, constant-factor approximation algorithm. We study the single vehicle problem with two hierarchical objectives. The first-level objective is to maximize the processed amount by a deadline, whereas the second-level objective is to minimize transportation costs. We propose two heuristic approaches to address the single vehicle problem. The first approach solves the MIP model with additional constraints to obtain feasible solutions with specific characteristics. The second approach is a prioritized bicriteria matheuristic that combines tabu search with linear programming. To evaluate the performance of these approaches, we provide an upper bounding scheme

on the processed amount by a deadline, and two relaxed MIP models to generate lower bounds on the transportation cost. The effectiveness of the proposed solution approaches is evaluated using realistic problem instances. Insights on key problem parameters and their effects on the solutions are extracted by further experiments.

ÖZETÇE

Sağlık hizmetlerinde, tanılayıcı ve tedavi edici önemli kararlar genellikle klinik testlerin sonuçlarına dayanır. Klinik laboratuvarlar, hastalardan toplanan numunelerin eşgüdümlü olarak toplanma, taşınma, ve işlenmesi ile testlerin zamanında ve hatasız gerçekleştirilmesini hedefler. Bu tezde, klinik laboratuvarlarda gözlenen numune toplama problemi ele alınmıştır. Bu problemde, coğrafi olarak dağınık lokasyonlarda zaman içerisinde biriken numunelerin sınırlı işleme kapasitesine sahip ekipmanla işlenmek üzere merkezi bir tesise taşınması gerekmektedir. Ortaya çıkan rotalama ve çizelgeleme problemi, işleme için toplama problemi (Collection for Processing Problem – CfPP) olarak adlandırılmıştır. Problemin başlıca özellikleri analiz edilmiş, doğrusal karma tamsayılı programlama (KTP) modeli geliştirilmiş, ve sezgisel çözüm yöntemleri tasarlanmıştır.

Çok araçlı problem tüm birimlerinin işlenmesinin bitiş süresinin enküçüklenmesi (makespan) amaç fonksiyonu ile çalışılmış ve polinom zamanlı, sabit faktörlü bir yaklaşıklama algoritması geliştirilmiştir. Tek araçlı problem iki sıradüzenli amaç fonksiyonu ile çalışılmıştır. Bu problemde, birincil amaç, verilen zaman sınırına kadar işlenen miktarı enbüyüklemek iken; ikincil amaç, taşıma maliyetlerini enküçüklemektir. Problemin çözümü için iki sezgisel yöntem geliştirilmiştir. İlk yöntem, bazı belirgin özelliklere sahip uygun çözümleri aramaya yönelik bir KTP modelinin çözülmesine dayanmaktadır. İkincisi ise, tabu arama yöntemi ile doğrusal programlamayı birleştiren önceliklendirilmiş iki kriterli bir sezgi ötesi yöntemdir. Önerilen sezgisel yöntemlerin performansını ölçmek amacıyla, zaman sınırına kadar işlenen miktar için bir üst sınır

bulma yöntemi ve taşıma maliyetleri için alt sınır bulan iki gevşetilmiş KTP modeli geliştirilmiştir. Aynı zamanda, önerilen sezgisel yöntemlerin etkinliği gerçekçi problem örnekleri üzerinde gösterilmiştir. Ayrıca, sayısal deneyler ile kilit problem parametreleri üzerinde öngörüler ve bunların çözümler üzerindeki etkileri çıkartılmıştır.

ACKNOWLEDGMENTS

I would first of all like to thank my supervisor, Dr. Sibel Salman for her great supervision and invaluable support during my Ph.D. study. I am also very thankful to Dr. Lerzan Örmeci and Dr. Esma Gel for their significant contributions to my thesis. They believed in me, shared their knowledge and experience with me, and paid attention to all the steps of my thesis. It was and will be a pleasure to work with them.

I thank Prof. Nesim Erkip for his sincere interest in my research and insightful comments and suggestions. I thank Dr. Metin Türkay for his helpful comments on my thesis. I also thank Dr. Necati Aras and Dr. Alper Yıldırım for being in my thesis committee despite the short time for the review of the thesis and for their comments.

I would like to thank my life long friends Dilek (Adak), Dilek (Uzun), Mehmet Ali, and Eren, for lifting my mood on the days when I have been feeling down, especially in the last year, and for making me believe that I can succeed. I thank Fadime (Üney Yüksektepe) for her close friendship and providing support. We shared the same anxiety and excitement with Zeynep (Turgay), Mehmet Can (Arslan), and Görkem (Sarıyer) as Ph.D. candidates. I thank them for their invaluable friendship.

All this would not be possible without the support of my family; my father, my mother, my brother, and my sister-in-law, Semra. I especially wish to thank them for their love, support, and sacrifice throughout my life. I would like to dedicate this dissertation to my husband, Ahmet, and our sons, Mert Selim and Mete Kerem. Without Ahmet's invaluable support, patience, and encouragement, it would be very difficult to complete this work. I thank and apologize to all three, who bore with me during this long and painful period.

This thesis was completed with the support of the TUBITAK scholarship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

CfPP	Collection for processing problem (single vehicle)
mCfPP	Multi vehicle collection for processing problem
MIP	Mixed integer programming
LP	Linear programming
VRP	Vehicle routing problem
TSP	Traveling salesman problem
Minmax m-TSP	TSP with m salesmen and makespan objective
TOP	Team orienteering problem
MTMCP	Multiple tour maximum collection problem
IRP	Inventory routing problem
VRP	Vehicle routing problem
VRPM	Vehicle routing problem with multiple tours
VRPTW	VRP with time windows
MOOP	Multi objective optimization problem
TF	Name of the proposed heuristic that solves an MIP model with a final TSP tour and the node filtering constraints
BM	Name of the proposed prioritized bicriteria matheuristic
TBH	Name of the proposed constructive myopic tour building heuristic, which is used to find the initial solution of BM
A-tour	after office-hour tour
D-tour	during office-hour tour

N	set of sites, where $N = \{1, 2, \dots, n\}$
Node 0	processing facility
N^+	set of nodes including the processing facility, i.e., $N^+ = N \cup \{0\}$
n	number of sites
A	set of arcs on N^+, i.e., $A = \{(i, j) \mid i, j \in N^+\}$
τ_e	time of day when sites close
τ_f	deadline for the processing of items
t_{ij}	traveling time from node i to node j, where $i, j \in N^+$
d_{ij}	distance from node i to node j, where $i, j \in N^+$
λ_i	accumulation rate at site i per unit time, where $i \in N$
μ	processing rate per unit time of the processing facility
κ_1	maximum number of tours that can be performed until time τ_e
κ_2	maximum number of tours that can be performed between time τ_e and τ_f
α_1	a workload parameter, the ratio of aggregated accumulation rate to the processing rate, $(\alpha_1 = (\sum_{i \in N} \lambda_i)/\mu)$
α_2	a workload parameter, the ratio of total accumulated amount to total processing capacity, $(\alpha_2 = (\tau_e \sum_{i \in N} \lambda_i)/(\tau_f \mu))$
D	total accumulated amount during $(0, \tau_e]$, i.e. $D = \sum_{i \in N} \lambda_i \tau_e$
$C_{\max}$	completion time of the processing of all the accumulated items
I	total idle time of the processor until all the accumulated items are processed, i.e., until $C_{\max}$
P_{τ_f}	processed amount by a due date τ_f
$\mathrm{UB}_{P(\tau_f)}$	an upper bound for the processed amount until τ_f

mCfPP(A)	Multi vehicle collection for processing problem with the objective A, where $A \in \{C_{\max}, I, P_{\tau_f}\}$
$\mathcal{N}(v)$	set of sites in cluster $v \in \{1, \ldots, m\}$
$\mathcal{D}(v)$	traveling time of the tour that starts and ends at node 0, and visits each site in $\mathcal{N}(v)$
$\Lambda(v)$	total accumulation rate of the sites in $\mathcal{N}(v)$, that is, $\Lambda(v) = \sum_{j \in \mathcal{N}(v)} \lambda_j$
δ^{mTSP}	approximation ratio for Minmax m-TSP
$\mathcal{G}$	set of feasible solutions for a CfPP instance
g	a solution (feasible scheduled giant-tour) in $\mathcal{G}$
$P^g(t)$	amount of items processed in time interval $(0, t]$, of solution g, where $t \in (0, \tau_f]$, $P^g(t) \in \mathbb{R}^+$
$C^g(t)$	transportation cost in time interval $(0, t]$, of solution g, where $t \in (0, \tau_f]$, $C^g(t) \in \mathbb{R}^+$
$C^g_{\max}$	maximum completion time (i.e. makespan) of processing of the total accumulated amount, of solution g, where $C^g_{\max} \in \mathbb{R}^+$
I^g	total idle time of the processor until all of the total accumulation is processed, i.e., until $C_{\max}$, where $I^g \in \mathbb{R}^+$
$\mathrm{UB}_{P(\tau_f)}$	upper bound for the processed amount by time τ_f, $\mathrm{UB}_{P(\tau_f)} \in \mathbb{R}^+$
$\mathrm{LB}_{C(\tau_f)}$	lower bound for the transportation cost, $\mathrm{LB}_{C(\tau_f)} \in \mathbb{R}^+$
R_k	return time of tour k, where $k = 1, 2, \ldots, \kappa_1 + \kappa_2$ and $R_k \in \mathbb{R}^+$
W_k	amount of waiting time before tour k, where $k = 1, 2, \ldots, \kappa_1 + \kappa_2$ and $W_k \in \mathbb{R}^+$
I_k	amount of idle time in the time interval $(R_k, R_{k+1}]$ where $k = 0, 1, \ldots, \kappa_1 + \kappa_2 - 1$ and $I_k \in \mathbb{R}^+$

$Q(t)$	number of items at the processing facility (either waiting in the queue or undergoing processing) at time t, where $t \in (0, \tau_f]$ and $Q(t) \in \mathbb{R}^+$ (Note that, we use Q_k to refer to $Q(R_k)$.)
T_{ik}	visit time of node i at tour k ($i \in N^+, k = 1, 2, \ldots, \kappa_1 + \kappa_2$). If node i is not visited in tour k, it denotes the last visit time of node i before tour k (i.e., $T_{ik} = T_{i,k-1}$). T_{0k} denotes the starting time of tour k.
π	current giant-tour
π^*	best found giant-tour
π'	neighboring giant-tour of giant-tour π, $\pi' \in \mathcal{N}(\pi)$
η	total number of iterations allowed
γ	number of iterations allowed without an improvement in any of the objective functions
θ	tabu tenure
K	number of tours performed up to time τ_e for the best found solution.
$\overline{v}$	percentage of sites that are visited, averaged over K tours for the best found solution (i.e., $100 \sum_{k=1}^{K} v_k/n$, where v_k is the number of sites visited in tour k.)
$\overline{f}$	percentage of sites that are filtered, averaged over K tours for the best found solution (i.e., $100 \sum_{k=1}^{K} m_k^*/n$, where the first m_k^* sites are filtered out in tour k.)
DA	achievement percentage, calculated as $100P(\tau_f)/(\tau_e \sum_{i \in N} \lambda_i)$.
DU	capacity utilization percentage, calculated as $100P(\tau_f)/(\mu\tau_f)$.
$\text{Gap}_{P(\tau_f)}$	percentage gap between $P(\tau_f)$ and $\text{UB}_{P(\tau_f)}$, calculated as $100(\text{UB}_{P(\tau_f)} - P(\tau_f))/P(\tau_f)$.

$\text{Gap}_{C(\tau_f)}$	percentage gap between $C(\tau_f)$ and $\text{LB}_{C(\tau_f)}$, where $\text{LB}_{C(\tau_f)}$ is the best bound found by CPLEX for the two relaxations at the end of the given run time limit (i.e., $100(C(\tau_f) - \text{LB}_{C(\tau_f)})/\text{LB}_{C(\tau_f)}$.
CPTU	Cost per 1,000 units processed

Chapter 1

INTRODUCTION

1.1 Motivation

Clinical laboratory testing is an essential element in the delivery of healthcare services. Physicians use laboratory tests to assist in the detection, diagnosis, evaluations, monitoring and treatment of diseases and other medical conditions. Clinical testing is performed on bodily fluids, such as blood and urine, and is usually outsourced to a specialized clinical testing company by hospitals and healthcare professionals. These companies analyze the specimens on special-purpose medical testing equipment, and compile the results for each patient in a report, which is then sent back to the patient's healthcare provider.

It is estimated that lab testing has an impact on over 70 percent of medical decisions [Knowledge Source Inc., 2008]. With the increasing awareness about early detection and prevention of diseases and the aging of the world population, the industry is poised for rapid growth [Koncept Analytics, 2008]. In fact, it is essential that clinical laboratories provide error-free processing, as well as reliable service with short turn-around times of test results.

Clinical testing companies are faced with the following daily operations. Clinical specimens are obtained from patients at various geographically dispersed locations referred to as collection sites, such as physician offices, hospitals, and patient service centers operated by the companies. The specimens are then transferred to a centralized processing facility to be analyzed. The collection of the specimens from the sites is achieved by a fleet of vehicles (and drivers) that visit each site a number of times every day. The collection sites are grouped into a number of geographical regions, each of which is serviced by one of the vehicles. The drivers visit the sites according to a predetermined sequence, except for situations when an urgent requisition requires them to expedite pick up of specimens from a specific site. Once the specimens get to the processing facility, the requisition orders are logged into a database system for tracking and billing of the orders. The specimens are then processed on a highly-automated testing equipment that runs at a predetermined rate of processing. Finally, the analysis results are compiled in a report. Figure 1.1 illustrates the specimen collection process.

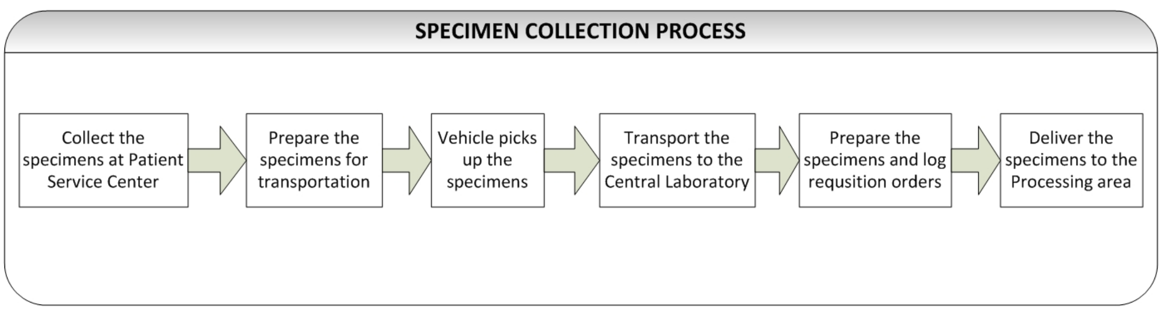


Figure 1.1: Process diagram for specimen collection

Most clinical testing companies have strict agreements on turn-around time, which is generally required to be less than one business day, and sometimes as short as a few hours for urgent requisitions. Customer satisfaction drops dramatically when a delay occurs in the delivery of the test results. Achieving a high level of customer service can be possible only if the collection and delivery of specimens to the laboratory is planned and executed efficiently.

Clinical testing companies are under increasing pressure to provide service at lower cost, and hence, they constantly seek ways to reduce their operational costs, even though their first priority is still to provide reliable and accurate results with a fast turn-around time. An important determinant of the operational costs is the utilization of the processing equipment and various other resources, such as labor, at the central laboratory. The processing equipment are generally complex, and have to be operated by highly-skilled labor. Hence, processing capacity on the equipment and labor resources have to be considered as bottleneck resources, and therefore, need to be effectively utilized.

An important factor in the utilization of the processing resources at the centralized laboratory is the way that the specimens are transported from the collection sites to the laboratory. In particular, the arrival rate of specimens to the centralized laboratory has to match (or exceed) the processing rate so that starvation of the processing resources can be avoided.

In this thesis, we consider the problem of transporting specimens from a number of geographically-dispersed collection sites to the centralized processing facility of a clinical testing company, and aim to provide generalized formulations and solution approaches for problems with similar characteristics from other domains. In particular, problems of concern have the characteristics of (i) collection of items from a number of geographically-dispersed nodes, where items accumulate over time, and (ii) dependency between the collection/transportation decisions and processing at centralized facilities.

There are other service systems with such challenging characteristics. One example is collection operations of small package pick-up/delivery companies. Similar to our problem, such operations require the collection of accumulated items at different locations and the subsequent processing of these items. Another example may arise in waste management operations, in the collection of hazardous materials and other waste that need to be processed and stored in a timely manner. In all these operations, the collection process is critical for the overall effectiveness of the system, since the utilization of the processing unit is a function of when and in what amounts the collected items arrive to the facility. In this thesis, the term collection for processing problem (CfPP) is used to describe the broad class of problems that are concerned with designing tours to match collected workload with the processing capacity at the processing facility.

In the context of our healthcare logistics problem, CfPP, the fundamental decisions to be optimized can be represented by the schedule (i.e., timing) of visits to each collection site throughout the day. Typically, this involves designing tours (sequence of sites to visit and their timing) to be conducted by a number of vehicles throughout the day. We note that this problem is significantly different from a traditional routing problem where the only decision of interest is the sequence in which the sites will be visited by each vehicle. Furthermore, collection strategies that only minimize the transportation cost may not match the processing rate, and cause excessive idle times at the central facility. On the other hand, keeping the processing unit busy requires more frequent, and possibly more costly tours. Hence, transportation decisions should be made with the consideration of the input that they will provide to the processing system. This requires significant extensions to the existing theory in this domain.

1.2 Solution methods

The optimization problem encountered in this thesis has the property that the set of feasible solutions might be so large even if the number of available vehicles and the number of collection sites are small. In such problems, simple exhaustive search methods require several days of computation even if fast computers are used.

Three types of solution methods are typically used to solve such problems (NP-hard problems):

- **Exact methods.** Exact algorithms are those that provide optimal solutions to the given problem, however, there is no guarantee on the running time of the algorithm [Woeginger, 2003]. We can not expect to develop exact algorithms that solve NP-hard problems in polynomial time, unless NP = P. As we prove that the problems studied in this thesis are NP-hard, we do not propose an exact method in this thesis.

- **Heuristics.** Heuristics are algorithms that find feasible solutions, obtained typically in polynomial time but have no guarantee on the quality of the solution [Lin and Kernighan, 1973]. The quality of the heuristics are typically tested empirically. Heuristics are generally used to solve real life problems because of their speed and their ability to handle large instances.

  A special class of heuristics that attracted a considerable amount of research attention in the last two decades is metaheuristics [Glover and Kochenberger, 2003]. Metaheuristics provide general frameworks for heuristics that can be applied to many problem classes. High solution quality is often obtained using metaheuristics for many combinatorial optimization problems, especially routing problems.

  In recent years, matheuristics that incorporate exact solvers in a heuristic context, both as primary solvers or as subprocedures, are an emerging field to solve complex real-world problems [Boschetti et al., 2009].

  In this thesis, we propose two matheuristic approaches to address the single vehicle CfPP. One of them is a heuristic approach that employs an exact solution procedure to a subproblem in a reduced solution space. The other one is a matheuristic that combines a metaheuristic algorithm with an exact solver to optimize the complementary decisions in a low-dimensional space in order to obtain candidate feasible solutions.

- **Approximation algorithms.** Approximation algorithms have a polynomial running time and return a feasible solution that is a certain factor away from the optimal solution for any instance of the problem [Dumitrescu and Mitchell, 2003]. An algorithm A for a minimization (resp. maximization) problem is said to be a δ-approximation algorithm if for every instance of the problem, A returns a solution with objective value at most δ (resp. at least $1/\delta$) times the optimal value. The parameter δ is called approximation guarantee or approximation ratio. The approximation ratio is always at least 1.

  In this thesis, we develop a constant factor approximation algorithm for the multi-vehicle CfPP.

1.3 Thesis contribution

This thesis introduces a new routing and scheduling problem with accumulating workload and processing capacity limitations. The problem has been inspired from an original practical logistics problem with significant economic impact in healthcare services provided by clinical laboratories. The problem is formalized as collection for processing problem (CfPP). CfPP captures the most important features of the real-world problem and defines a relevant modeling for the problem.

In the study of the multi-vehicle problem with makespan objective, we provide the first approximation result and show that a constant-factor polynomial time approximation exists for this strongly NP-hard problem. The proposed approximation algorithm is based on the idea of partitioning the sites into clusters, and assigning a single vehicle and a portion of the total processing capacity that is proportional to the aggregated accumulation rate of the cluster to each one. Hence, the algorithm solves a single vehicle problem for each cluster. Since assigning a single vehicle to a dedicated region of sites is a practically used solution approach in the real-world problem and the worst-case bound of the proposed algorithm seems to be promising, we study the single vehicle problem.

In the study of the single vehicle problem with two hierarchical objectives, a far-from-trivial MIP formulation is derived with the help of some simple but important properties and two heuristic approaches are proposed to address the problem. The first heuristic handles two objectives in a lexicographic approach and solves an MIP model with additional constraints that reduce the solution space. We demonstrate the effectiveness of the proposed solution approach using realistic problem instances, where the problem size (the number of nodes) varies between 10 and 18. Experiments extensively evaluate the solution approach and the impact of different parameters, both from a computational and a practical point of view. However, since this approach is based on solving an MIP formulation, it is not suitable for larger problem instances. In order to address larger instances, we propose a second heuristic for the problem.

The second approach is a novel prioritized bicriteria matheuristic (BM) that combines tabu search with linear programming (LP). BM determines the node sequence of tours through a tabu search algorithm and schedules the node visit times optimally through a linear program. The major difference of our approach from a classical tabu search algorithm lies in: (1) how it handles two prioritized objectives in guiding the search, and (2) how it effectively combines two methods for optimizing the routing and scheduling decisions consecutively. The computational experiments on realistic test instances, with number of nodes varying between 10 and 51, show that BM finds reasonably good solutions even for relatively large instances.

Another contribution of the thesis on the single vehicle problem is providing methods to bound the two objectives. A simple polynomial-time algorithm that calculates a strong upper bound on the processed amount by a deadline is proposed. For the challenging task of generating strong lower bounds on the transportation cost for a given processed amount, we derive valid inequalities and develop two relaxed MIP models.

Finally, although the thesis problem is described in the context of a healthcare logistics problem, the proposed approaches are expected to find application in other industries involved in pick up, delivery, and subsequent processing of accumulating items, such as package pick-up/delivery companies. Hence, the research represents a broader agenda that is likely to impact several other industries and software companies that provide decision support systems for routing and scheduling.

1.4 Thesis outline

This thesis contains seven chapters. Chapter 2 provides a general literature review related to the collection for processing problem (CfPP). Since the problem has not been studied before, to the best of our knowledge, in Chapter 2, we review the studies on specimen collection and the relevant research that possess similar characteristics with CfPP, as well as a brief explanation of how our problem is different from the previously considered problems in the literature.

Chapter 3 studies the multi-vehicle problem, mCfPP. We show that this problem with the makespan objective (mCfPP(Cmax)) is NP-hard and develop a polynomial-time, constant-factor approximation algorithm for mCfPP(Cmax). The problem with a single site is addressed as a special case to identify the minimum number of vehicles to achieve the lower bound on the makespan and to analyze the optimal makespan for a single.

Chapter 4 defines the single vehicle problem with two hierarchical objectives: (i) maximizing the processed amount by a deadline, such as the next morning, and (ii) minimizing the transportation cost. We formulate a linear mixed integer programming (MIP) model to solve the bicriteria problem in two levels. We provide an upper bounding scheme on the processed amount by a deadline, and develop two relaxed MIP models to generate lower bounds on the transportation cost. We show that the problem is NP-hard and provide two heuristic approaches in Chapters 5 and 6. Chapter 5 proposes a heuristic approach based on solving the MIP model with additional constraints that seeks feasible solutions with specific characteristics. The effectiveness of the proposed solution approach is evaluated using realistic problem instances. Chapter 6 describes a prioritized bicriteria matheuristic (BM) that combines an exact linear programming method with a tabu search metaheuristic. Computational study on realistic problem instances (with up to 51 nodes) shows that BM is superior to the MIP based heuristic of Chapter 5 in terms of both solution quality and computation time.

The thesis is concluded with short summary, conclusions, and future research directions in Chapter 7.

Chapter 2

LITERATURE REVIEW

The core problem studied in this thesis, *collection for processing problem*, is a vehicle routing problem with multiple tours, accumulation over time, and a finite processing capacity. To the best of our knowledge, this problem has not yet been studied in the literature. In this chapter, we, first, provide the two studies on clinical specimen collection and their differences from this research in Section 2.1. Next, we review four classes of the well-known routing problems that possess similar characteristics with the collection for processing problem. These problems are traveling salesman problem and its related variants (Section 2.2), team orienteering problem (Section 2.3), vehicle routing problem and its related variants (Section 2.4), and inventory routing problem (Section 2.5). Finally, we provide the distinctive features of the collection for processing problem from the current literature in Section 2.6.

2.1 Studies on clinical specimen collection

In the literature, to the best of our knowledge, there are only two studies that address the specimen collection problem that arise in clinical laboratories. The first one is the case study of McDonald [1972]. Considering a fixed number of vehicles, the objective of this study is to minimize total traveling time to make all collections subject to a constraint on traveling time. With this objective, the author aims to solve a vehicle scheduling problem with an algorithm that uses the savings algorithm in conjunction with a simultaneous route building algorithm. In McDonald [1972], the problem is formulated as a Vehicle Routing Problem (VRP), rather than a multi-tour problem.

Revere [2004] also discusses a case study on a business process re-engineering project for a laboratory courier service. The goal is to minimize both the laboratory courier and staffing costs. The problem is divided into two sub-problems each dedicated to one of the two objectives. For the problem with the objective of minimizing laboratory courier cost, the set-partitioning traveling salesman model is used to determine the courier routes that meet the customer service demands. However, since in the case of Revere [2004] there exists a time limit of two hours for a tour, sub-tours, which return to the processing facility without visiting all sites, should be allowed. For this reason, the linear programming model for traveling salesman problem, with manually created sub-tours is used as the solution approach. For the second problem with the objective of minimizing the staffing cost, an integer programming formulation is developed. The drawback of their approach is that the manual creation of traveling salesman sub-tours can only be applied to problems where the number of nodes are relatively few and the desired routes are constant. In addition, the integer programming formulation for the staffing problem can be applicable for small size problems with the assumption that employees should always be responsible for the same route.

In contrast with our study, neither of these studies considers the accumulation of specimens over time.

2.2 Traveling salesman problem and its related variants

One of the simplest, but still NP-hard, routing problems is the traveling salesman problem (TSP). The aim of TSP is to determine a tour that starts from a base city, visits all other given cities exactly once, and terminates at the base city with the objective of minimizing the total distance traveled. Let t_{ij} be the distance from city i to city j. If $t_{ij} = t_{ji}$ for all i and j, then the traveling salesman problem is referred to as a symmetric traveling salesman problem (STSP); otherwise, it is called an asymmetric traveling salesman problem (ATSP). The problem is said to be Euclidean if the distance between two cities is the Euclidean distance. If the distances satisfy the triangular inequality, i.e., if $t_{ij} \leq t_{ik} + t_{kj}$ for all i, j and k, then the problem is known as metric TSP.

The TSP is one of the most studied NP-hard problems in the literature and solution methods for this problem have reached a very high level. Very large Euclidean instances of the TSP can be solved to optimality, the largest instance solved to optimality so far contains 24,978 cities. It was solved by branch-and-cut by the research team of Applegate, Bixby, Chvatal, Cook and Helsgaun[1]. Heuristic methods for the TSP have been applied to an instance with 1,904,711 cities throughout the world and the gap between the currently best know upper and lower bounds for this instance has been shown to be 0.0477%[2]. More general routing problems like the problem of this research turn out to be much harder to solve, both heuristically and exactly, compared to the TSP.

Traveling salesman problem with profits and m-traveling salesman problem are two variants of the TSP that are related with the collection for processing problem. In the following subsections, a brief introduction of the problems with relevant literature is provided.

[1] http://www.tsp.gatech.edu/sweden/index.html

[2] http://www.tsp.gatech.edu/world/index.html

2.2.1 Traveling salesman problem with profits

The traveling salesman problem with profits can be considered as a bicriteria TSP with two different objectives, maximizing the collected profit and minimizing the traveling cost. Solving TSPs with profits should result in finding a noninferior solution set, i.e., a set of feasible solutions such that neither objective can be improved without deteriorating the other [Feillet et al., 2005]. According to the way the two objectives are addressed, the literature on the TSP with profits can be classified into three categories: (i) Both objectives are combined in the objective function; the aim is to find a circuit that minimizes travel costs minus collected profit. This type of problem is defined as the profitable tour problem (PTP) by DellAmico et al. [1995] and did not attract so much attention in the literature. (ii) The travel cost objective is stated as a constraint, the aim is to find a circuit that maximizes collected profit such that travel costs (time) do not exceed a preset value. This type of problem is called the orienteering problem (OP) [Chao et al., 1996] and studied extensively in the literature. (iii) The profit objective is stated as a constraint, the aim is to find a circuit that minimizes travel costs and whose collected profit is not smaller than a preset value. This type of problem is called the prize-collecting TSP (PCTSP) by Balas [1989] and studied extensively in the literature. For an extensive survey on TSPs with profits, the reader is referred to Feillet et al. [2005].

2.2.2 m-Traveling salesman problem

The m-traveling salesman problem (m-TSP) is a generalization of the TSP that introduces more than one salesman. In the m-TSP, there are n cities, m salesmen and one depot. All cities should be visited exactly once on one of m tours, starting and ending at the depot. The tours should visit at least one city. If the distances satisfy the triangle inequality, i.e., if $t_{ik} \leq t_{ij} + t_{jk}$ for all i, j and k, then it is easy to see that the distance of the shortest TSP tour on the n cities plus the depot is always less than or equal to the distance of the shortest m-TSP solution for any m. Any m-TSP with n cities can be formulated as a TSP with $m + n$ cities. One first creates m copies of the depot node. The distances between depot nodes is then set to a sufficiently large number while the distances between the depot nodes and ordinary nodes are copied from the m-TSP. The large distance between depot nodes ensures that no salesmen tours are empty. Notice that the resulting TSP does not obey the triangle inequality. The m-TSP is not studied widely in the literature, probably because it is so closely related to the TSP. The literature about heuristics and exact methods has recently been surveyed by Bektas [2006].

An interesting variant of the problem is the MinMax m-TSP where the length of the longest salesman tour has to be minimized. Frederickson et al. [1978] proves that the problem is NP-complete for $k \geq 2$ and provides two approximation algorithms for metric MinMax m-TSP, i.e., MinMax m-TSP with cost matrix, C, satisfying the triangle inequality. The first heuristic of Frederickson et al. [1978] constructs m tours simultaneously by using a least cost insertion criterion or nearest neighbor criterion. The authors show that the worst case performance ratio of this heuristic is equal to two for the least cost insertion criterion, and $m + (m/2) \log n$ for the nearest neighbor criterion. The second heuristic of Frederickson et al. [1978], first, obtains a good TSP solution and then splits the TSP tour into m subtours of more or less equal costs. The worst case performance ratio of this heuristic is equal to $b + 1 - 1/m$, where b is a worst case bound for the single TSP algorithm. Thus, if Christofides' algorithm [Christofides, 1976] is used to construct the TSP tour, b takes the value $3/2$ and the worst case performance ratio of the second heuristic is equal to $5/2 - 1/m$. As the time complexity of Christofides' algorithm is $O(n^3)$, this is a polynomial time approximation algorithm. Franca et al. [1995] describes a tabu search heuristic and two exact search schemes for MinMax m-TSP. The proposed algorithms can solve problems involving up to 50 nodes to optimality. Applegate et al. [2002] solved a challenging MinMax m-TSP instance to optimality for the first time. The instance originated from a competition from 1996 and had been unsolved since then. The problem was solved on a network of 188 processors and required 10 days of computing, which corresponds to roughly 79×106 CPU seconds scaled to a 500 MHz Alpha EV6 processor.

2.3 Team orienteering problem

The team orienteering problem (TOP) aims to route a number of vehicles through a set of nodes each of which contains a fixed reward (i.e., no accumulation by time), in order to maximize the total collected reward while ensuring that all vehicles return to the pre-determined nodes within a given time limit. Multiple Tour Maximum Collection Problem (MTMCP) has the same objective but the tours are required to start and end at same node.

Two exact algorithms have been proposed for the TOP; one that uses column generation [Butt and Ryan, 1999] and another one that uses a branch-and-price algorithm [Boussier et al., 2007], to solve small- to moderate-sized instances with up to 100 vertices, provided the number of vertices in each tour remains relatively small.

Brideau and Cavalier [1994] proposed a greedy construction procedure for the MTMCP. Campbell et al. [1998] developed a 5-step metaheuristic to solve TOP, whose structure was based on a deterministic variant of simulated annealing [proposed in Dueck and Scheuer, 1990]. Tang and Miller-Hooks [2005] designed a tabu search heuristic for TOP that outperformed other existing heuristic techniques based on experiments conducted on a set of benchmark problems. A survey of heuristic methods is provided in Vansteenwegen et al. [2009].

The first study in the field of TOP and MTMCP that considers time-dependent rewards is by Tang et al. [2007]. Tang et al. [2007] formulates a variant of TOP with time-dependent rewards and proposes a tabu search heuristic to solve the problem.

2.4 Vehicle routing problem and its related variants

In the classical vehicle routing problem (VRP), a fleet of identical vehicles must be routed to serve a known and finite set of customers with known demands and return to a fixed depot at the end of each tour. The assumptions of the classical VRP can be summarized as follows. In each tour at least one customer must be visited and each customer must be served by exactly one vehicle. If the vehicle has a capacity limit, then the total amount of demand of the customers on a tour can not exceed the vehicle capacity, Q. If there exists a maximum tour time constraint, then the duration of a tour should not exceed the time limit, L. Each vehicle can be assigned at most one tour. Generally problems are defined with symmetric cost matrix. Variants with asymmetric cost matrix are explicitly denoted (e.g., AVRP, etc.). General objectives of extensively studied VRP are: (1) minimizing fixed costs for used vehicles plus transportation costs (distance traveled) for a fixed fleet size, (2) minimizing number of vehicles, (3) balancing of the tours, for travel time and vehicle load, and (4) minimizing the penalties for partially served customers. A recent and extensive survey on exact and heuristic methods to solve the VRP can be found in Cordeau et al. [2007].

Many variations of VRP have been widely studied in the literature. Vehicle routing problem with multiple tours and vehicle routing problem with time windows are two variants of the VRP that are related with the collection for processing problem. In the following subsections, a brief introduction of the problems with relevant literature is provided.

2.4.1 Vehicle routing problem with multiple tours

In the classical VRP, each vehicle can be used at most once over the planning period. However, the assumption that a single vehicle can perform only one tour in VRP is very limiting, since multiple uses of a vehicle might result in significant cost savings. In vehicle routing problem with multiple tours (VRPM), it is possible to assign more than one tours to the same vehicle over the planning period. Specifically, VRPM aims to determine a set of routes and an assignment of each route to one vehicle, which minimizes the total routing costs and satisfies (i) each route starts and ends at the depot, (ii) each customer is visited by exactly one route, (iii) the demand of the customers in the same route does not exceed Q, and (iv) the duration of routes assigned to the same vehicle does not exceed T.

Although in practice multiple tour assignment is common, VRPM is not studied widely in the literature. Fleischmann [1990] is the first study that incorporates the multi-tour idea into the VRP. Later, Taillard et al. [1996] proposes a population based algorithm using tabu search and bin packing approaches. In this algorithm, first, a set of VRP solutions is constructed from a population of routes generated using the tabu search heuristic and then, bin-packing is used to allocate routes to vehicles. Golden et al. [1996] adopts this approach to solve a similar VRPM using a minimax objective (minimizing the length of the maximum distance covered by any vehicle at the end of the day). Their approach is applicable to a wide range of VRP variants. Brandao and Mercer [1997, 1998] study real life applications of VRPM, which require more assumptions such as vehicles with heterogenous capacities, maximum legal driving time per day for drivers, rules that restrict to access customers, and etc. They propose a tabu search algorithm for a realistic case. Later, Petch and Salhi [2004] propose a multi-phase construction heuristic for VRPM, and Salhi and Petch [2007] develop a genetic algorithm for VRPM. Their objective is to minimize the maximum overtime restriction for a prescribed number of vehicles. Olivera and Viera [2007] develop an adaptive memory procedure for VRPM, in which initial VRP solutions are constructed by the sweep algorithm which are then enhanced by a tabu search. Alonso et al. [2008] study the site-dependent multi-tour periodic vehicle routing problem and develop a mathematical model and a tabu search procedure.

2.4.2 Vehicle routing problem with time windows

The vehicle routing problem with time windows (VRPTW) is a generalization of the classical VRP where the service at each customer must start within an associated time window and the vehicle must remain at the customer location during service. The VRPTW arises in many real-world applications such as in bank deliveries, postal deliveries, industrial refuse collection, national franchise restaurant services, school bus routing, industrial gases delivery, furniture deliveries, and JIT (just in time) manufacturing [Braysy and Gendreau, 2002]. According to hard time windows, the vehicle cannot arrive at a node after the latest time to begin service and it must wait if it arrives at customer i before time a_i. However, soft time windows can be violated at a cost. The research on VRPTW mainly focus on hard time windows. There exists a vehicle capacity limit Q, which is the maximum amount of demand that can be carried by a vehicle.

For recent surveys on the state of the art in VRPTW research the reader is referred to Cordeau et al. [2002] that describes both exact and heuristic methods, the survey of Braysy and Gendreau [2005a,b] that describes metaheuristics and the survey of Kallehauge [2008] that provides formulations and exact algorithms.

Next, we review two real life applications of the VRPTW, which are closely related with the collection for processing problem. The first one arises in the process of blood collection and the second one occurs in a lean production system.

Blood collection process

In the blood collection process, vehicles collect blood at prescheduled sites for approximately six hours. Whole blood is picked up during the collection and brought to the center for processing. According to the corresponding regulations, the blood must be processed within eight hours after donation to extract platelets. Collection for processing problem has three key common features with blood collection problems studied before: (i) units to be collected accumulate over time, (ii) collected units should be processed by a processing center, and (iii) there is a time concern on the processing of collected units.

Operational efficiency in blood collection in supply centers without vehicle routing has been studied since 70s [e.g., Dumas and Rabinowitz, 1977, Or, 1976]. Yi and Scheller-Wolf [2003] is the first paper that incorporates the vehicle routing aspects into this problem, through a real life application in American Red Cross (ARC). Since a vehicle should visit a site during the collection hours according to the blood collection time regulations, the problem is modeled in the class of VRP with time windows (VRPTW). The difference of this problem from the VRPTW is threefold: (i) not all sites are required to be visited, (ii) goods are continuously produced at each site during a certain production time window, and (iii) the amount collected from a site depends on the arrival/departure times of the vehicle. Yi and Scheller-Wolf [2003] proposes a four-step solution procedure for the problem. In the first step, all of the feasible routes are generated. In the second step, the arrival/departure times for each route is optimized in order to maximize the amounts collected in each tour. In the third step, a large proportion of inferior routes are removed from the feasible set. Finally, an integer programming model is used to determine the best subset of routes. Their algorithm is applied in ARC for one month and it is observed that their solution resulted in nearly 60% (on average) reduction in the total traveling distance compared to the used policy in practice. In this study, only a single pickup per campaign event is considered, therefore the interdependence between time windows is not considered.

Recently, Doerner et al. [2008] studied logistics of blood program in Austrian Red Cross as a vehicle routing problem with multiple interdependent time windows (VRPmiTW) for each customer. Similar to the case in Yi and Scheller-Wolf [2003], according to the regulations of Eastern Austria, all blood needs to be processed in one blood bank within about 5 hours after donation due to processing requirements. The goal is to find minimum cost tours and to allocate the appropriate transport devices to take back all the blood. The critical issue that defines the interdependency in time windows in this problem is that the dispatching decisions made in the solution procedure directly influence the time windows for visits at the campaign events. More precisely, since the vehicle might not necessarily return to the blood bank after visiting a site, the pickups at several sites should be combined on one tour in a way that each pickup event should obey the time regulations. Since the interdependencies of the multiple time windows at the customer locations had not been studied up to that time, the modeling of this novel problem is stated as a contribution of the paper. After providing a complete mixed-integer programming model formulation for the problem, the authors proposed several variants of a constructive heuristic as well as an exact branch-and-bound algorithm to solve the VRPmiTW. Both approaches are based on the savings idea [Clarke and Wright, 1964]. The computational analysis was performed on 22 realistic data sets. For each problem instance, the number of sites varies between 6 and 15, and the number of required pickups at each site ranges from 0 (short campaigns where the team takes all the donated blood to the blood bank at the end of the campaign) to 4. The computational results showed that the heuristic techniques produce solutions reasonably close to the optimal solutions in negligible time for a basic case, where the number of pickups is fixed at the theoretical minimum number of pickups for each site. If the number of pickups for selected sites is increased, high additional cost reductions are observed. In such cases, although the computational times for the exact method are prohibitively large, the proposed heuristics provide very good solutions within seconds.

Although the studies on blood collection consider accumulation over time in the sites, none of them considers the impact of finite processing capacity.

Lean production systems

The inbound logistics problems arising in just-in-time or lean production systems are concerned with coordinating the material in-flow with the production rate, similar to the CfPP. Such studies concentrate on the coordination of routing with production.

In a recent study, Ohlmann et al. [2008] consider vehicle routing with time-windows to collect components from suppliers in a lean production system. The objective is to minimize the total traveling cost subject to inventory limitations. They consider multiple vehicles and equally-spaced visits to suppliers by different vehicles. A vehicle is allowed to perform only one tour. Because of the hardness of the problem, they present a two-phase routing and scheduling approach based on a nested tabu search heuristic. Our study differs from Ohlmann et al. [2008] in that CfPP allows multiple uses of the same vehicle, which is also called route linking in the literature. As stated in Ohlmann et al. [2008] incorporation of route linking significantly complicates the overall problem. In addition, different from the problem of Ohlmann et al. [2008], our problem does not require equally spaced and equal sized deliveries and does not have capacity and volume restrictions and time-window constraints.

2.5 Inventory routing problem

A principal stream of literature that combines the considerations of inventory control and vehicle routing is the inventory routing problem (IRP). The IRP is concerned with the distribution of products from a single facility to a set of nodes to satisfy customer demand over a given planning horizon. The objective of IRP is to minimize the operating costs, which consists of transportation and inventory holding costs. An overview of the problem and a survey of early research can be found in Moin and Salhi [2007]. IRP constructs routes for the distribution vehicles in such a way that shortages are avoided while unnecessary deliveries are not scheduled.

Both IRP and the collection for processing problem combine a temporal element (the time at which deliveries are done) with a spatial element (the routing of vehicles), but there is no consideration of processing in IRP. The IRP can be considered as a dual counterpart of our problem: (1) the nodes of IRP consume the units brought to them at a constant rate, as opposed to the sites of our problem which accumulate the items at a constant rate, and (2) the shortage is defined at the nodes in IRP, whereas in our problem a shortage may occur at the processing center. Different from the IRP, in our problem, accumulated items do not incur holding costs. To summarize, although these two problems share a number of key features, it is not possible to formulate one in terms of the other.

2.6 Distinctive features of the collection for processing problem

While the problems described in the previous sections of this chapter have similarities to the collection for processing problem, the basic differences of our problem from these problems can be summarized as follows.

First, our problem has a scheduling aspect. Since the items accumulate at the sites over time, the timing of the visits affects the collected amounts from the sites. Therefore, different from the traditional routing problems such as VRP, TSP, and their variants, not only the sequence of sites to be visited by each vehicle, but also their timing should be determined in this problem.

Second, the transferred items are input to the processor, so that the transportation decisions also affect the idleness of the processor, which in turn affects the completion time of the processing of all the accumulated amount. Therefore, collection strategies that only minimize the transportation cost may result in starvation of the processing resources and late delivery of test results. Hence, routing and scheduling decisions should be made with the consideration of the processing capacity and the accumulated amounts. This is significantly different from the existing literature in this domain.

Third, the vehicle tours have to be designed simultaneously due to the significant dependency between them. In every tour, the return time of the tour to the processing facility, as well as the collected amount affect the processed amount up to the return time of the next tour. Furthermore, due to accumulation, the amount collected from a node in a tour depends on the last visit time of that node.

Chapter 3

A CONSTANT-FACTOR APPROXIMATION ALGORITHM FOR MULTI-VEHICLE PROBLEM WITH MAKESPAN OBJECTIVE

3.1 Introduction

This chapter studies the multi-vehicle collection for processing problem, mCfPP, in which the items that accumulate at a number of sites during $(0, \tau_e]$ should be transported by a number of uncapacitated vehicles to the processing facility with a limited processing capacity. Multiple uses of the same vehicle is allowed. There are three natural objectives in mCfPP: minimizing maximum completion time of processing of all items (i.e. makespan), minimizing total idle time of the processor, or maximizing the processed amount by a deadline. In this chapter, after proving that minimizing makespan is equivalent to minimizing total idle time of the processor and the solution that minimizes makespan also maximizes the processed amount by any deadline, we focus on the problem with makespan objective. We prove that the problem is NP-hard through an approximation preserving reduction from a two-stage, hybrid flowshop scheduling problem. We analyze two special cases of the problem: single site, multi-vehicle and single site, single vehicle. For the first one, we identify the necessary number of vehicles to achieve a lower bound on the makespan, and for the second one, we characterize the optimal makespan under different workload settings. With insights obtained from these cases, we develop a constant-factor approximation algorithm for the general multi-site, multi-vehicle case. The algorithm forms disjoint clusters of the sites and assigns a vehicle to each cluster. Each vehicle repeats the same tour over time.

The remainder of this chapter is organized as follows. Section 3.2 describes the problem. Section 3.3 analyzes the problem with a single site. Section 3.4 presents the proposed approximation algorithm. Section 3.5 concludes with a summary and contributions of this chapter.

3.2 Problem description

The problem is defined on a directed graph $G = (N^+, A)$ with node set $N^+ = \{0, 1, 2, \ldots, n\}$ and arc set $A = \{(i, j) \mid i, j \in N^+\}$. Node 0 represents the processing facility where m identical and uncapacitated vehicles are stationed. $N = N^+ \setminus \{0\}$ represents the set of sites from which items should be collected for processing. Items accumulate at each site $i \in N$ at a constant and known rate of λ_i units per unit time between time 0 and time τ_e. All the accumulated items should be processed at the processing facility, which has a constant and known processing rate of μ items per unit time. The items are assumed to be identical, requiring equal amount of processing time, without loss of generality. The accumulated items should be collected and transferred to a centralized processing facility via a set of tours performed by the m vehicles. Each arc $(i, j) \in A$ is associated with a non-negative travel time t_{ij}. The traveling times are assumed to be metric. The collection and processing can start at time 0, and a vehicle may perform more than one tour. Since accumulation of items ends at time τ_e, a feasible solution should visit all sites at least once after time τ_e to collect any accumulated items remaining. In the clinical testing context, the items represent the specimens.

We define a measure of the total workload at the processing facility as $\alpha_1 = \sum_{i \in N} \lambda_i / \mu$. We let $D = \sum_{i \in N} \lambda_i \tau_e$ denote the total accumulated amount during

$(0, \tau_e]$ and $Q(t)$ the number of items at the processing facility (either waiting in the queue or undergoing processing) at time t. We set $Q(0) = q_0$.

We consider three different objectives for the problem:

- Minimizing the completion time of the processing of all the accumulated items (i.e., makespan), which is denoted by $C_{\max}$.

- Minimizing the total idle time of the processor until all the accumulated items are processed, i.e., until $C_{\max}$, which is denoted by I.

- Maximizing the processed amount by a due date τ_f (where τ_f is sufficiently large to visit all sites after τ_e), which is denoted by P_{τ_f}. (This objective is equivalent to minimizing the total number of tardy items.)

As the items are identical, their processing order does not affect any of these objectives. Therefore, for these objectives, the problem is to construct and schedule a set of tours for each vehicle. Each tour starts and ends at node 0 and includes visits to a subset of sites in N. We denote the problem with objective A, where $A \in \{C_{\max}, I, P_{\tau_f}\}$, as mCfPP($A$). Propositions 3.2.1 and 3.2.2 show that mCfPP(I) and mCfPP($C_{\max}$) are equivalent, and an optimal solution to mCfPP($C_{\max}$) is also optimal for mCfPP(P_{τ_f}), respectively. Therefore, we focus on mCfPP($C_{\max}$) in the remainder of the paper.

Proposition 3.2.1. *The problems mCfPP(I) and mCfPP($C_{\max}$) are equivalent.*

Proof. We first note that both problems have the same feasible set. Given a feasible solution, we have:

$$C_{\max} = I + D/\mu, \tag{3.1}$$

where D/μ is the total processing time. Since D and μ are constant parameters of the problem, minimization of the total idle time is equivalent to minimization of the makespan. □

Proposition 3.2.2. *Any optimal solution of mCfPP($C_{\max}$) is also optimal for mCfPP(P_{τ_f}) but any optimal solution of mCfPP(P_{τ_f}) is not necessarily optimal for mCfPP($C_{\max}$).*

Proof. Suppose that we have optimal solutions to mCfPP($C_{\max}$) and mCfPP(P_{τ_f}) denoted by s^C and s^P, respectively. There are two possible cases based on whether s^C can process all the total accumulated amount within the given due date:

(*i*) $C_{\max}(s^C) \leq \tau_f$ meaning that $P_{\tau_f}(s^C) = D$, and

(*ii*) $C_{\max}(s^C) > \tau_f$ meaning that $P_{\tau_f}(s^C) < D$.

In the case $C_{\max}(s^C) \leq \tau_f$, clearly, s^C is optimal for mCfPP(P_{τ_f}) as no items are tardy. Let us consider the case $C_{\max}(s^C) > \tau_f$. Assume s^C is not optimal for mCfPP(P_{τ_f}). Then, $P_{\tau_f}(s^C) < P_{\tau_f}(s^P)$ and the amount of unprocessed items by time τ_f in solution s^C is strictly larger than that amount in s^P. Therefore, at time τ_f, it requires more time to process the remaining unprocessed items in solution s^C, which contradicts with the optimality of s^C for mCfPP($C_{\max}$).

To show that s^P is not necessarily optimal for mCfPP($C_{\max}$), let us consider the case $C_{\max}(s^C) \leq \tau_f$. Then, $P_{\tau_f}(s^C) = P_{\tau_f}(s^P) = D$. However, s^P might process some items in the time interval $(C_{\max}(s^C), \tau_f]$ so that the makespan of s^P is strictly larger than that of s^C. □

3.2.1 NP-hardness of mCfPP($C_{\max}$)

In order to prove that mCfPP($C_{\max}$) is NP-hard, we use an approximation preserving reduction from an extended version of the two-stage, hybrid flowshop scheduling problem with m parallel machines (processors, the words are used interchangeably) in the first stage and a single machine in the second stage. This scheduling problem is referred to as $FH2,(Pm^{(1)},1^{(2)})||C_{\max}$, according to the notation used by Ahmadi et al. [1992]. In $FH2,(Pm^{(1)},1^{(2)})||C_{\max}$, there are n jobs to be processed in two stages, where each job i should be processed first in stage 1 with processing time p_i, and then in stage 2 with processing time q_i. The objective is to minimize the makespan.

We next define an extended version of $FH2,(Pm^{(1)},1^{(2)})||C_{\max}$, denoted by $FH2,(Pm^{(1)},1^{(2)}),(\beta \to \gamma)||C_{\max}$, where β denotes the stage with batch processors, γ denotes the stage with a single discrete processor, and $\to$ denotes the flow-shop configuration. We will show that $FH2,(Pm^{(1)},1^{(2)}),(\beta \to \gamma)||C_{\max}$ reduces to mCfPP($C_{\max}$). This problem extends $FH2,(Pm^{(1)},1^{(2)})||C_{\max}$ by allowing batch availability and batch setup times in all processors of stage 1. A constant setup time, s, is incurred whenever a batch is formed on any processor in stage 1. Transfer of jobs to the second stage is possible only when processing of all jobs belonging to the same batch is completed in the first stage. The problem is to determine the optimal batch composition and to schedule the jobs in two stages so that the makespan is minimized.

Lemma 3.2.1. *$FH2,(Pm^{(1)},1^{(2)}),(\beta \to \gamma)||C_{\max}$ is NP-hard.*

Proof. Although the two machine flow shop problem with the objective of minimizing makespan, (i.e., $F2||C_{\max}$) is polynomially solvable, its extended version with batch availability and batch setup times in the first stage (i.e., $F2(\beta \to \gamma)||C_{\max}$) was shown to be strongly NP-hard [Cheng and Wang, 1998]. $F2(\beta \to \gamma)||C_{\max}$ is a special case of $FH2,(Pm^{(1)},1^{(2)}),(\beta \to \gamma)||C_{\max}$ since it has a single processor in the first stage. Therefore, $FH2,(Pm^{(1)},1^{(2)}),(\beta \to \gamma)||C_{\max}$ is strongly NP-hard. □

The NP-hardness of mCfPP($C_{\max}$) is implied by the following proposition.

Lemma 3.2.2. *There is an approximation preserving, polynomial-time reduction from* $FH2,(Pm^{(1)},1^{(2)}),(\beta \to \gamma)||C_{\max}$ *to mCfPP($C_{\max}$).*

Proof. First, note that m vehicles of an mCfPP($C_{\max}$) instance correspond to m machines of the first stage and the processing facility of the mCfPP($C_{\max}$) instance corresponds to the single machine in the second stage in an instance of $FH2,(Pm^{(1)},1^{(2)}),(\beta \to \gamma)||C_{\max}$, which we refer to as the scheduling problem, or SP, in short from here on.

Given an instance I_1 of the SP with processing times p_i and q_i for all $i \in \{1,2,\ldots,n\}$, and s, we construct an instance I_2 of the mCfPP($C_{\max}$) consisting of n sites (where $N^+ = \{0,1,\ldots,n\}$ and $N = N^+ \setminus \{0\}$) as follows. For each site $i \in N$, we set $t_{ij} = p_i$, for all $j \in N^+, t_{0i} = s$, and $\lambda_i = q_i/\tau_e$. We also set $\mu = 1$ and $\tau_e = s$ so that the accumulation ends at the earliest possible visiting time of any site. Clearly, the construction of mCfPP($C_{\max}$) instance takes polynomial time.

Suppose that there exists a feasible SP solution to I_1 with makespan $C^*_{\max}$. We construct an mCfPP($C_{\max}$) solution to I_2 as follows. As vehicles correspond to machines in the SP instance, the tours of a vehicle correspond to the batches scheduled on the corresponding machine in the first stage. For each tour of a vehicle, the sites are traveled in the order of jobs scheduled in the corresponding batch. Then, the duration of a tour is equal to the processing time of the batch plus the setup time by construction. Furthermore, the amount collected by a tour becomes available for processing at the end of the tour. As each job is scheduled in the first stage in the SP solution, all sites are visited by a vehicle. In addition, as $\tau_e = s$ and $t_{0i} = s$, all the accumulated amount at each site i, i.e., $\lambda_i\tau_e = q_i$, is collected by a visit. Then, the processing time of the collected amount from a site is equal to the processing time of the corresponding job in the second stage. Since all jobs must be processed in the second stage, all the accumulated amount is processed. Therefore, we obtain a feasible solution for instance I_2 of the mCfPP($C_{\max}$) with makespan $C^*_{\max}$.

Next, let us consider a feasible mCfPP($C_{\max}$) solution to the constructed instance I_2 with makespan $C^*_{\max}$. We obtain an SP solution to I_1 as follows. For each vehicle, we assign a distinct machine in stage 1. Then, we form a batch from each tour of the vehicle. We schedule each batch in the order of the tours and schedule the jobs in the batch according to the order of the sites visited in the tour. Processing time of the batch will be equal to the duration of the tour. Also, note that each job must be scheduled once since each site is visited once. As all items should be processed by the processing facility, all jobs are processed in the second stage. Since there is no batch processing in the second stage, the scheduling of the jobs in the second stage might be done according to the FCFS (first-come, first-served) order. Therefore, we obtain a feasible solution for instance I_1 of the SP with makespan $C^*_{\max}$.

As a result, an optimal solution to I_2 with makespan $C^*_{\max}$ yields an optimal solution to I_1 with the same makespan value.

□

Lemmas 3.2.1 and 3.2.2 lead to Proposition 3.2.3.

Proposition 3.2.3. *mCfPP($C_{\max}$) (or mCfPP(I), equivalently) is strongly NP-hard.*

3.2.2 Preliminaries

A simple lower bound on the optimal makespan, $C^*_{\max}$, follows from Equation (3.1) as $C^*_{\max} \geq D/\mu = \alpha_1 \tau_e$. This bound shows how makespan increases with the workload parameter. When $\alpha_1 < 1$, we can tighten the bound as $C^*_{\max} \geq \tau_e$ since the accumulation at sites ends at time τ_e. $C^*_{\max} \geq \tau_e$ can be further improved by using the fact that in a feasible solution, the sites should be visited after τ_e to collect the remaining items. Then, the smallest time to transfer all items to the processing facility is $\tau_e + \max_{i \in N} t_{i0}$. Thus,

$$C^*_{\max} \geq \max\{\alpha_1 \tau_e, \tau_e + \max_{i \in N} t_{i0}\}. \tag{3.2}$$

The makespan may increase further by the idleness of the processor. Next, we analyze the conditions under which the processing facility remains idle in $[0, C_{\max}]$. Suppose that we have a feasible solution with κ tours. We index the tours in the order of their arrival time to the processing facility. Hence, tour $k \in \{1, 2, \ldots, \kappa_s\}$ corresponds to the k^{th} arriving tour. R_k is the return time of tour k to the processing facility. Note that, by definition, $R_{k-1} \leq R_k$. Figure 3.1 illustrates the two possible cases based on whether idleness occurs between two consecutive tour return times, R_k and R_{k+1}.

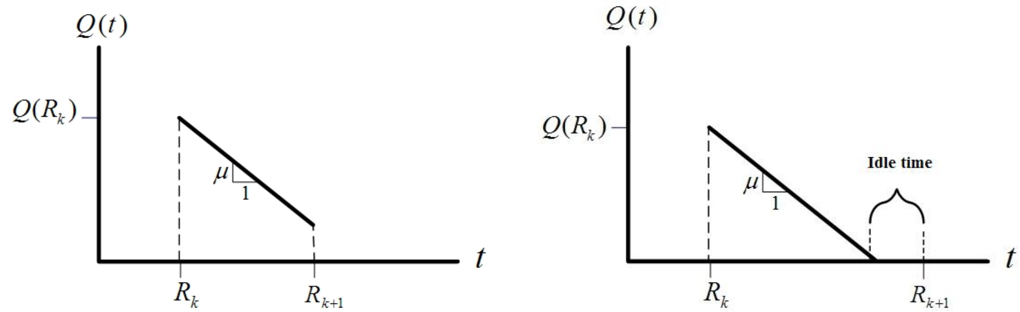
Let us define $R_0 = 0$ and $R_{\kappa+1} = C_{\max}$ for ease of notation, and I_k as the idle time in the time interval $(R_k, R_{k+1}]$ for $k = 0, 1, \ldots, \kappa$. Then, I_k depends on (i) the queue level at R_k, i.e., $Q(R_k)$, and (ii) $R_{k+1} - R_k$. If $Q(R_k) \geq \mu(R_{k+1} - R_k)$, then the processor does not remain idle during $(R_k, R_{k+1}]$ as in Figure 3.1a. Otherwise, as in Figure 3.1b, the processor idles for $I_k = (R_{k+1} - R_k) - Q(R_k)/\mu$ time units.

If $Q(0) = 0$, then the processor is idle during $(0, R_1]$. That is, $I_0 = R_1$. Since all items should be delivered by time R_κ, the processor does not idle in $(R_\kappa, C_{\max}]$, i.e., $I_\kappa = 0$. Therefore, minimizing the idleness of the processor in $(0, R_\kappa]$ is sufficient to minimize the total idle time.

3.3 Special case: mCfPP with a single site

In this section, we analyze the simple case with a single collection site. In Section 3.3.1, we determine the necessary and sufficient number of vehicles to achieve the lower bound on $C^*_{\max}$ and in Section 3.3.2, allowing only a single vehicle, we characterize $C^*_{\max}$. As $C_{\max} = I + \alpha_1 \tau_e$, we minimize either $C_{\max}$ or I during the analysis. The impact of using a single vehicle on the makespan in a single-site system is analyzed in Section 3.3.3.

Let us consider the case where the set N consists of a single site with accumulation



(a) $I_k = 0$ **(b)** $I_k = (R_{k+1} - R_k) - Q(R_k)/\mu$

Figure 3.1: Idle time between return times of two consecutive tours. The figures (a) and (b) show cases without idleness and with idleness, respectively.

rate λ and traveling time t to and from the processing facility. We assume that $\tau_e > 2t$; that is, at least one tour can be completed by time τ_e, and the processor queue is initially empty, i.e., $Q(0) = 0$. In addition, we assume that at least one item accumulates and can be processed in t time units, i.e., $\lambda \geq 1/t$ and $\mu \geq 1/t$, where $t \geq 1$. In this setting, $\alpha_1 = \lambda/\mu$.

For this setting, first, the bound of $C^*_{\max} \geq \alpha_1 \tau_e$ can be improved based on the assumption that the processor queue is initially empty. Since the processor is certainly idle in the time interval $(0, R_1]$ and duration of a trip to the site is $2t$, we can write $C^*_{\max} \geq 2t + \alpha_1 \tau_e$. Second, the bound of $C^*_{\max} \geq \tau_e + t$ can be improved if we assume that there is at least one time unit between consecutive visits to the site. Then, since the processing of the items transferred the last requires at least $\lambda/\mu = \alpha_1$ time units, $C^*_{\max} \geq \tau_e + t + \alpha_1$. As a result, we get

$$C^*_{\max} \geq \max\{2t + \alpha_1 \tau_e, \;\; \tau_e + t + \alpha_1\}. \tag{3.3}$$

3.3.1 Multiple vehicles

In this section, we determine the necessary and sufficient number of vehicles, denoted by m^*, to achieve the lower bound on $C^*_{\max}$ given in Equation (3.3). For this purpose, we differentiate between two cases depending on the workload parameter, $\alpha_1 < 1$ and $\alpha_1 \geq 1$. In each case, we will construct the tours consecutively, starting from the first tour, and identify the required number of vehicles for each case.

Case 1 ($\alpha_1 \geq 1$): When $\alpha_1 \geq 1$, Equation (3.3) gives $2t + \alpha_1\tau_e \geq \tau_e + t + \alpha_1$, so that $C^*_{\max} = 2t + \alpha_1\tau_e$. In a solution that achieves $C^*_{\max} = 2t + \alpha_1\tau_e$, the first tour should be completed at time $R_1 = 2t$ and the processor should not idle after R_1. Now, setting $R_1 = 2t$, let us construct the consecutive tours to avoid idleness. Then, tour $k = 2, \ldots, \kappa$ is completed at time $R_k = R_{k-1} + \alpha_1^{k-1}t$. We set the number of tours, i.e., κ, to the minimum integer that satisfies $2t + \sum_{k=1}^{\kappa} \alpha_1^k t \geq \tau_e + t$ in order to guarantee that there are no unnecessary tours and tour κ visits the site after τ_e. Let A_k be the collected amount in tour k. Since $A_1 = \lambda t$, and $A_k = \lambda(R_k - R_{k-1})$ for $k = 2, \ldots, \kappa - 1$, the processor never idles after R_1, i.e., $C^*_{\max} = 2t + \alpha_1\tau_e$ is achieved, in this solution. Figure 3.2 shows the timeline representation of the constructed tours. Note that, as $\alpha_1 \geq 1$, $R_k - R_{k-1}$ gets larger with k.

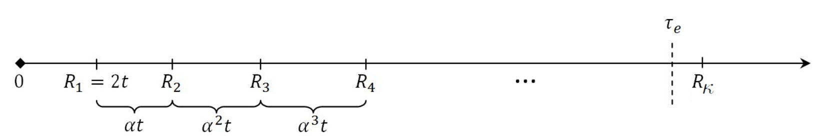


Figure 3.2: Timeline representation of tours constructed for Case 1 ($\alpha_1 \geq 1$)

Now let us determine the number of vehicles required in this solution. When $\alpha_1 \geq 2$, $R_k - R_{k-1} = \alpha_1^{k-1}t$ is greater than or equal to a tour duration, i.e., $R_k - R_{k-1} \geq 2t$

for all k so that a single vehicle is sufficient to perform all tours in order to achieve $C^*_{\max} = 2t + \alpha_1\tau_e$, i.e., $m^* = 1$. Note that the vehicle waits to match the required R_k values.

When $1 \leq \alpha_1 < 2$, a single vehicle is not sufficient to prevent idleness after $R_1 = 2t$ since the earliest return time of the second tour is $R_1 + 2t = 4t$ in any solution, and the collected amount by the first tour, which is equal to λt, is completely depleted at $R_1 + \lambda t/\mu = 2t + \alpha_1 t$, which is strictly smaller than $4t$. We now demonstrate that two vehicles are sufficient. As $R_3 - R_1 = \alpha_1 t + \alpha_1^2 t \geq 2t$ and $R_k - R_{k-1}$ gets larger as k gets larger, the odd tours in Figure 3.2 can be performed by one vehicle and even tours can be performed by another vehicle. Therefore, when $1 \leq \alpha_1 < 2$, two vehicles are necessary and sufficient (that is, $m^* = 2$) to achieve $C^*_{\max} = 2t + \alpha_1\tau_e$.

Case 2 ($\alpha_1 < 1$): When $\alpha_1 < 1$, we set $C^*_{\max}$ to the larger of $2t+\alpha_1\tau_e$ and $\tau_e+t+\alpha_1$ based on the values of α_1, τ_e, and t. Then, $I^* = C^*_{\max} - \alpha_1\tau_e$. Similar to Case 1, we can construct a solution with $R_1 = I^*$ and prevent idleness after R_1. Now, contrary to Case 1, as $\alpha_1 < 1$, $R_k - R_{k-1}$ gets smaller as k gets larger. Therefore, the required number of vehicles, m, in this solution, is equal to the number of tours performed during $(R_\kappa - 2t, R_\kappa]$ as a vehicle tour requires $2t$ time units. Before analyzing the value of m, we should note that m gives only an upper bound on m^*. (Note that in Case 1 we calculate the exact value of m^*.) The reason is that there might be another optimal solution with a smaller number of vehicles, which has $R_1 < I^*$ and allows a total idle time of $I^* - R_1$ after R_1.

Let us find an upper bound on the number of tours performed in interval $(R_\kappa - 2t, R_\kappa]$, denoted by $\overline{m}$, which is also an upper bound on m^*. As we assume that there is at least one time unit between consecutive visits of the site, the first tour of $(R_\kappa - 2t, R_\kappa]$, say tour k', collects at least λ items. It requires $\lambda/\mu = \alpha_1$ time units to process this amount. Therefore, tour $k'+1$, must return before $R_{k'} + \alpha_1$. It collects at most $\alpha_1\lambda$ items, which requires a processing time of $\alpha_1\lambda/\mu = \alpha_1^2$ time units. Therefore, tour $k'+2$, must return before $R_{k'+1}+\alpha_1^2$. In a similar fashion, we find that $\overline{m}$ is the largest integer that satisfies $\sum_{k=1}^{\overline{m}} \alpha_1^k \leq 2t$.

We summarize the results of this section in the next proposition.

Proposition 3.3.1. *If m^* is the number of vehicles necessary and sufficient to achieve $C^*_{\max} = \max\{2t+\alpha_1\tau_e,\ \ \tau_e+t+\alpha_1\}$, then,*

$$m^* = \begin{cases} 1, & \text{if } \alpha_1 \geq 2; \\ 2, & \text{if } 1 \leq \alpha_1 < 2; \\ \leq \overline{m}, \ \text{where } \overline{m} \text{ is the largest} & \\ \text{integer that satisfies } \sum_{k=1}^{\overline{m}} \alpha_1^k < 2t, & \text{if } 0 < \alpha_1 < 1. \end{cases}$$

3.3.2 Single vehicle

Now let us analyze $C^*_{\max}$ when there is only one vehicle.

Case 1 ($\alpha_1 \geq 2$): According to Proposition 3.3.1, if $\alpha_1 \geq 2$, a single vehicle is sufficient to achieve the lower bound, i.e., $C^*_{\max} = 2t + \alpha_1\tau_e$.

Case 2 ($1 \leq \alpha_1 < 2$): If $1 \leq \alpha_1 < 2$, we can construct a feasible solution in which the vehicle performs κ tours where tour $k = 1, \ldots, \kappa$ is completed at time $R_k = 2tk + w$, where $w \geq 0$ is the waiting time of the vehicle at the processing facility before starting the first tour. We will determine the optimal level of w in order to minimize the idleness. In this solution, the first tour collects $A_1 = \lambda(t+w)$ items and each tour $k = 2, \ldots, \kappa-1$ collects $A_k = 2\lambda t$ items, where $R_k - R_{k-1} = 2t$, for $k = 1, \ldots, \kappa$. Since $\alpha_1 \geq 1$ and $Q(R_k) \geq A_k = 2\lambda t \geq \mu(R_{k+1} - R_k) = 2\mu t$, the processor never stays idle after R_2 as in the case of Figure 3.1a. Now let us analyze the amount of idle time in $(0, R_2]$, which is equal to $I_1 + I_2$. Since $I_1 = R_1 = 2t + w$

and

$$\begin{aligned} I_2 &= \max\{0, R_2 - R_1 - A_1/\mu\} \\ &= \max\{0, 2t - \lambda(t+w)/\mu\} \\ &= \max\{0, 2t - \alpha_1(t+w)\}, \end{aligned}$$

we have $I_1 + I_2 = \max\{2t + w, t(4 - \alpha_1) + w(1 - \alpha_1)\}$.

$I_1 + I_2$ is minimized when $w = t(2/\alpha_1 - 1)$. Setting $w = t(2/\alpha_1 - 1)$ gives a makespan of $C^*_{\max} = \alpha_1 \tau_e + (1 + 2/\alpha_1)t$.

Case 3 ($\alpha_1 < 1$): When $\alpha_1 < 1$, let us construct a feasible CfPP solution, in which the vehicle performs $\kappa = \lfloor(\tau_e + t)/(2t)\rfloor$ tours, where tour $k = 1, \ldots, \kappa$ is completed at time $2tk + w$ and tour κ visits the site exactly at time τ_e. That is, $R_\kappa = \tau_e + t$ and $w = \tau_e + t - 2t\kappa$ is the waiting time of the vehicle at the processing facility before starting the first tour. Note that, $w < 2t$ since $\kappa = \lfloor(\tau_e + t)/(2t)\rfloor$. In this solution, the first tour collects $A_1 = \lambda(t+w)$ items and each tour $k = 2, \ldots, \kappa$ collects $A_k = 2\lambda t$ items and satisfies $R_{k+1} - R_k = 2t$.

In order to calculate the makespan of this solution, let us analyze whether all items collected by the first $\kappa - 1$ tours have been processed by the end of tour κ, i.e., $R_\kappa = \tau_e + t$. Let us assume that in case of an arrival of new items, these items are processed first, that is, newly arriving items have higher priority than the items in the processor queue, if any. Note that this does not affect the makespan but simplifies the following analysis. For the first tour, $A_1 = \lambda(t+w) = \lambda(\tau_e + 2t(1-\kappa))$. Then, all items collected in the first tour can be processed by the time R_2, if $A_1 \leq 2\mu t$, which requires $\alpha_1 \leq 1/(\tau_e/2t + 1 - \kappa)$. Otherwise, we have $A_1 - 2\mu t$ items waiting in the queue at R_2. For tours $k = 2, \ldots, \kappa - 1$, as $A_k/\mu < R_{k+1} - R_k$, all items collected in a tour are processed by the end of the next tour by our assumption. In addition, there is an excess processing capacity of $\mu(R_{k+1} - R_k) - A_k = 2t(\mu - \lambda)$, which can be used to process some of the items collected by the first tour. Therefore, the amount that is waiting in the queue at R_2 can be processed by R_κ, if $A_1 - 2\mu t \leq 2t(\kappa-2)(\mu-\lambda)$, which requires $\alpha_1 \leq (\kappa-1)/(\tau_e/2t-1)$. Then, if $\alpha_1 \leq (\kappa-1)/(\tau_e/2t-1)$, this solution has a makespan of $C_{\max} = R_\kappa + A_\kappa/\mu = \tau_e + (1+2\alpha_1)t$. In order to see the optimality of this solution, let us compare it with a solution that visits the site at $\tau_e + x$, where $x > 0$. In this solution, the last tour is completed at $\tau_e + x + t$ and collects at least $\max\{1,\ \lambda(2t-x)\}$ items. It takes at least $\max\{1/\mu,\ \alpha_1(2t-x)\}$ time units to process that amount. Therefore, this solution has a makespan of $\tau_e + x + t + \max\{1/\mu,\ \alpha_1(2t-x)\}$, which is strictly larger. Therefore, the constructed solution is optimal for the case $\alpha_1 \leq (\kappa-1)/(\tau_e/2t-1)$ where $\alpha_1 < 1$.

For $(\kappa-1)/(\tau_e/2t-1) < \alpha_1 < 1$, we provide an upper bound on $C^*_{\max}$ since finding $C^*_{\max}$ is cumbersome. Let us analyze the solution constructed in the previous paragraphs for this case. Note that, $A_1 < \lambda 3t$ since $w < 2t$. As $\mu 2t$ of A_1 is processed during $(R_1, R_2]$ and some of the remaining unprocessed items collected in the first tour can be processed up to R_κ, the amount waiting in the queue at R_κ is strictly less than $A_\kappa + (A_1 - \mu 2t) = 5\lambda t - 2\mu t$. As $R_\kappa = \tau_e + t$, we get $C^*_{\max} < R_\kappa + (5\lambda t - 2\mu t)/\mu = \tau_e + (5\alpha_1 - 1)t$. Same as the case with $\alpha_1 \leq (\kappa-1)/(\tau_e/2t-1)$, a solution cannot have a makespan smaller than $R_\kappa + A_\kappa/\mu$, which is greater than or equal to $\tau_e + (1+2\alpha_1)t$. Thus, $\tau_e + (1+2\alpha_1)t \leq C^*_{\max} < \tau_e + (5\alpha_1 - 1)t$.

Proposition 3.3.2 summarizes the analysis.

Proposition 3.3.2. *When there is a single vehicle, if*

(i) $\alpha_1 \geq 2$, *then* $C^*_{\max} = 2t + \alpha_1 \tau_e$*;*

(ii) $1 \leq \alpha_1 < 2$, *then* $C^*_{\max} = \alpha_1 \tau_e + (1 + 2/\alpha_1)t$*;*

(iii) $\alpha_1 < 1$ *with* $\alpha_1 \leq (\kappa-1)/(\tau_e/2t-1)$, $C^*_{\max} = \tau_e + (1+2\alpha_1)t$*;*

(iv) $\alpha_1 < 1$ *with* $\alpha_1 > (\kappa-1)/(\tau_e/2t-1)$, $\tau_e + (1+2\alpha_1)t \leq C^*_{\max} < \tau_e + (5\alpha_1 - 1)t$.

3.3.3 Analysis of the results

We now analyze the impact of using a single vehicle on the makespan in a single-site system by comparing the results in Propositions 3.3.1 and 3.3.2.

According to Proposition 3.3.1, for $\alpha_1 \geq 2$, a single vehicle is sufficient to obtain the optimal makespan. For $\alpha_1 < 2$, let us compare the optimal makespans given in Propositions 3.3.1 and 3.3.2. The difference between the optimal makespans of single and multi-vehicle cases is at most t, when $1 \leq \alpha_1 < 2$, and at most $3t$, when $\alpha_1 < 1$. Therefore, using a single vehicle increases the optimal makespan by at most $3t$ time units. This shows the importance of the distance of the site to the processing facility. When t is relatively small, utilizing a single vehicle does not worsen the makespan significantly. Depending on the operating cost of the vehicle, this may even be preferable to using more vehicles.

We can utilize the results of Propositions 3.3.1 and 3.3.2 in developing a heuristic solution approach for the general case with multiple sites as follows. First, let us consider the case with a single vehicle to serve multiple sites and obtain a solution for this case by transforming the problem to a single site problem. Note that in the corresponding single site problem, the collection rate of the site should be the sum of the collection rates of all sites. Then, according to the workload of the problem, we can construct a solution as in Section 3.3.2 for this single site problem. When we implement this solution with multiple sites, we schedule the vehicle to make identical tours that visit all sites in a short time, by solving a Traveling Salesman Problem (TSP) to near-optimality. Next, let us consider the case with multiple vehicles. Now we can partition the sites into clusters, and assign a single vehicle to each one. If the clusters are formed such that the sites within a cluster can be toured in a short time, then we expect the makespan to be reasonably low. This idea leads to the approximation algorithm provided in Section 3.4.

3.4 Solution approach

In this section, we develop a polynomial time algorithm for mCfPP($C_{\max}$) that produces provably good solutions for every instance. The algorithm is composed of two phases. In the first phase, called *Clustering*, the set of sites is partitioned into m disjoint subsets where m is the number of vehicles. In the second phase, called *Scheduling*, the touring schedule of each vehicle is determined. Each vehicle is assigned to exactly one subset, and performs tours to transfer the accumulated workload of the sites in the subset to the processing facility. We benefit from the tour construction of Section 3.3.2 in this phase. A description of each phase is provided in Sections 3.4.1 and 3.4.2, respectively.

An algorithm is said to be a $\delta-$approximation algorithm (or have a δ approximation ratio) for a minimization problem P, if for any instance of P, it yields a feasible solution whose objective function value is at most δ times the optimum value. For NP-hard problems, developing polynomial time approximation algorithms is a widely accepted approach to cope with computational difficulty while providing worst-case performance guarantees [Williamson and Shmoys, 2010]. In Section 3.4.3, we prove that our proposed algorithm has a constant approximation ratio.

3.4.1 Clustering

The aim of the clustering phase is to form m disjoint clusters of sites such that the maximum traveling time within the clusters is minimized. We define the traveling time of a cluster as the shortest traveling time of a tour that starts and ends at node 0 and visits each site in the cluster exactly once.

The clustering problem we want to solve is equivalent to the k-Traveling Salesman Problem (k-TSP, $k \geq 2$) defined by Frederickson et al. [1978], and also referred to as Minmax m-TSP by Franca et al. [1995]. In Minmax m-TSP, an undirected complete graph $G = (N, A)$ is given with a distinguished initial node 0 and a symmetric matrix of nonnegative integer costs (distances or travel times) $C = (c_{ij})$ associated with A. There are m identical vehicles based at node 0. A solution consists of m vehicle tours starting and ending at node 0 such that every node $N \backslash \{0\}$ is visited by one vehicle. The objective is to minimize the maximum cost of the m tours. Frederickson et al. [1978] prove that the problem is NP-hard for $k \geq 2$ and provide two approximation algorithms for metric Minmax m-TSP (i.e., the cost matrix, C, satisfies the triangle inequality). The first algorithm constructs m tours simultaneously. The authors show that the worst-case performance ratio of this heuristic is $\delta = 2$, if the least cost insertion criterion is used for tour construction. The second algorithm of Frederickson et al. [1978] first obtains a good TSP solution and then splits the TSP tour into m subtours of more or less equal costs. The approximation ratio of this algorithm is equal to $\delta = b + 1 - 1/m$, where b is the worst-case ratio for the TSP algorithm. Thus, if Christofides' algorithm [Christofides, 1976] is used to construct the TSP tour, $b = 3/2$ and $\delta = 5/2 - 1/m$. Franca et al. [1995] describe a tabu search heuristic and two exact search schemes for Minmax m-TSP that can solve problems involving up to 50 nodes to optimality.

In the clustering phase, we use the first heuristic of Frederickson et al. [1978], which runs in polynomial time and is the best approximation algorithm proposed so far having an approximation ratio of two. In general, any other approximation algorithm for Minmax m-TSP, with performance ratio δ^{mTSP}, can be utilized.

Given a solution by this algorithm, let $\mathcal{N}(v) \subseteq N$ represent the set of sites in cluster $v \in \{1, \ldots, m\}$, and let $\mathcal{A}(v) \subseteq A$ represent the set of edges between the nodes in $\mathcal{N}(v) \cup \{0\}$. For this approximate Minmax m-TSP solution, let $\mathcal{D}(v)$ represent the traveling time of the tour that starts and ends at node 0, and visits each site in $\mathcal{N}(v)$, and let $\Lambda(v)$ represent the total accumulation rate of the sites in $\mathcal{N}(v)$, that is, $\Lambda(v) = \sum_{j \in \mathcal{N}(v)} \lambda_j$. Then, $\max_{v=1,\ldots,m} \mathcal{D}(v) \leq \delta^{mTSP} D_m^*$, where D_m^* is the optimal value of the Minmax m-TSP problem.

3.4.2 Scheduling

In the scheduling phase, we assign a single vehicle to each cluster and then construct and schedule the tours of each vehicle, independently of each other. Construction of the tours of each vehicle is inspired by the single vehicle, single site case provided in Section 3.3.2. In Section 3.3.2, we showed that a single vehicle performing subsequent tours without waiting in between them prevents the idleness of the processor after the second tour when $\alpha_1 \geq 1$, and minimizes the unprocessed amount at time τ_e when $\alpha_1 < 1$. Based on this observation, the scheduling phase makes each vehicle perform tours without waiting. Each tour of a vehicle is the same as the tour of the corresponding vehicle in the Minmax $m-$TSP solution. Then, for each cluster $v \in \{1, \ldots, m\}$, starting with time 0, the vehicle performs $\kappa(v)$ identical tours, each with traveling time $\mathcal{D}(v)$. Clearly, tour $k = 1, \ldots, \kappa(v)$ of cluster v is completed at time $R_k = \mathcal{D}(v)k$, that is, $R_{k+1} - R_k = \mathcal{D}(v)$. Note that the starting time of tour $\kappa(v)$ is $R_{\kappa(v)-1} = \mathcal{D}(v)(\kappa(v) - 1)$. In order to prevent unnecessary tours, we set $\kappa(v)$ to be the minimum integer that satisfies $\mathcal{D}(v)(\kappa(v) - 1) \geq \tau_e$ so that all sites are visited after τ_e. Then, for cluster v,

- the first tour collects less than $\Lambda(v)\mathcal{D}(v)$,
- each tour $k = 2, \ldots, \kappa(v) - 2$ collects exactly $\Lambda(v)\mathcal{D}(v)$ items,
- tour $\kappa(v) - 1$ collects less than or equal to $\Lambda(v)\mathcal{D}(v)$ items as some of the sites might be visited after time τ_e, and
- tour $\kappa(v)$ collects an amount strictly less than $\Lambda(v)\mathcal{D}(v)$ as all sites are visited after time τ_e.

3.4.3 Performance analysis

We will derive an upper bound on the makespan of solutions generated by our algorithm. In order to facilitate the analysis, let us assume that each cluster has a dedicated processor with processing rate $\mu(v) = \mu\Lambda(v)/\sum_{i\in N}\lambda_i = \Lambda(v)/\alpha_1$ so that $\sum_{v=1,\ldots,m}\mu(v) = \mu$. As $\Lambda(v)/\mu(v) = \alpha_1$, the workload level of the whole system is preserved in each one of the clusters.

Let us consider a solution output by the algorithm. For each cluster v, in order to find the total idle time of its dedicated processor, $I(v)$, and the makespan, $C_{\max}(v)$, we can perform an analysis based on the workload level, α_1.

- **Case 1** ($\alpha_1 \geq 1$): The processor never stays idle in $(R_2, R_{\kappa-1}]$, since $Q(R_k) \geq \Lambda(v)\mathcal{D}(v) \geq \mu(v)(R_{k+1} - R_k)$ for $k = 2, \ldots, \kappa(v) - 2$. However, the processor is idle during the first tour and idleness might occur during tours two and $\kappa(v)$. As a result, the total idle time of this processor is at most three times the tour length, that is, $I(v) \leq 3\mathcal{D}(v)$. Therefore, $C_{\max}(v) \leq \Lambda(v)\tau_e/\mu(v) + 3\mathcal{D}(v) = \alpha_1\tau_e + 3\mathcal{D}(v)$.

- **Case 2** ($\alpha_1 < 1$): Idleness occurs during every tour, since $Q(R_k) = \Lambda(v)\mathcal{D}(v) \leq \mu(v)(R_{k+1} - R_k)$. This means that all the items delivered so far are processed by the end of each tour. As the last tour collects an amount strictly less than $\Lambda(v)\mathcal{D}(v)$, $C_{\max}(v) < R_\kappa + \Lambda(v)\mathcal{D}(v)/\mu(v) = R_\kappa + \alpha_1\mathcal{D}(v) < R_\kappa + \mathcal{D}(v)$. Since $R_\kappa \leq \tau_e + 2\mathcal{D}(v)$, $C_{\max}(v) \leq \tau_e + 3\mathcal{D}(v)$.

Let $C_{\max}$ be the makespan of the solution found by the algorithm. If $\alpha_1 \geq 1$, then $C_{\max} \leq \alpha_1\tau_e + 3\max_{v=1,\ldots,m}\mathcal{D}(v) = \alpha_1\tau_e + 3\delta^{mTSP}D^*_m$. Otherwise, $C_{\max} \leq \tau_e + 3\max_{v=1,\ldots,m}\mathcal{D}(v) = \tau_e + 3\delta^{mTSP}D^*_m$. That is,

$$C_{\max} \leq \max\{1, \alpha_1\}\tau_e + 3\delta^{mTSP}D^*_m. \tag{3.4}$$

In order to find the approximation ratio of our algorithm, we may compare this upper bound with the lower bound of $C^*_{\max} \geq \max\{\alpha_1\tau_e, \tau_e\}$ provided in Equation (3.2). However, when $\alpha_1 < 1$, $C^*_{\max} \geq \tau_e$ provides a loose bound since it does not consider the necessity to visit all sites after τ_e. Therefore, we can improve this bound by adding the minimum time to visit all sites using m vehicles to τ_e. If the problem of finding this time is referred to as problem P, and T^* denotes the optimal objective value of P, then $C^*_{\max} \geq \tau_e + T^*$. In problem P, a vehicle may not be at node 0 at time τ_e, that is, it may be on a tour that started earlier. Therefore, problem P resembles Minmax m-TSP, except that all vehicles are at node 0 at time 0 in Minmax m-TSP, whereas vehicles are not required to be at node 0 at τ_e in P. Accordingly, the optimal value of the Minmax m-TSP, D^*_m, can be used to find a lower bound for P.

Let us first obtain a Minmax m-TSP solution from a P solution with objective value T. A P solution is composed of paths for m vehicles, where a path starts at a node in $\mathcal{N}(v) \cup \{0\}$ or at a point on an edge in $\mathcal{A}(v)$, visits a subset of sites in $\mathcal{N}(v)$, and returns to node 0. The paths that start at node 0 are, in fact, tours and we keep them in the Minmax m-TSP solution. The remaining paths are converted to a tour as follows. We delete the part of the path up to its first node (say, node i), if any, and add edge $(0, i)$ to this path. By this procedure we obtain a Minmax m-TSP solution with objective value of at most $T + \max_{i \in N} t_{0i}$. Since D^*_m is the optimal value of the Minmax m-TSP, $T + \max_{i \in N} t_{0i} \geq D^*_m$. Then, $T \geq D^*_m - \max_{i \in N} t_{0i}$.

Since we have metric t_{ij}, $\max_{i \in N} t_{0i} \leq D^*_m/2$. Therefore, $T \geq D^*_m/2$. Then,

$$C^*_{\max} \geq \max\{\alpha_1\tau_e, \tau_e + D^*_m/2\}. \tag{3.5}$$

Having this stronger bound, we can show that our algorithm has a constant approximation factor.

Theorem 3.4.1. *Our clustering-based algorithm is a* $(6\delta^{mTSP} + c)$*-approximation algorithm for mCfPP($C_{\max}$), where* δ^{mTSP} *is the approximation ratio for Minmax* *m-TSP and*

$$c = \begin{cases} 1, & \text{if } \alpha_1 > 6\delta^{mTSP} \\ 0, & \text{otherwise.} \end{cases}$$

Proof. Let us analyze the two cases of $\alpha_1 < 1$ and $\alpha_1 \geq 1$. If $\alpha_1 < 1$, Equation (3.4) gives $C_{\max} \leq \tau_e + 3\delta^{mTSP} D_m^* \leq 6\delta^{mTSP}(\tau_e + D_m^*/2)$. Since $D_m^* \leq 2(C_{\max}^* - \tau_e)$ due to Equation (3.5), $C_{\max} \leq 6\delta^{mTSP} C_{\max}^* - (6\delta^{mTSP} - 1)\tau_e$. As $\delta^{mTSP} \geq 1$, $6\delta^{mTSP} - 1 \geq 1$ giving that $C_{\max} \leq 6\delta^{mTSP} C_{\max}^*$.

If $1 \leq \alpha_1 \leq 6\delta^{mTSP}$, Equation (3.4) gives $C_{\max} \leq \alpha_1 \tau_e + 3\delta^{mTSP} D_m^*$. Since $D_m^* \leq 2(C_{\max}^* - \tau_e)$ due to Equation (3.5), $C_{\max} \leq 6\delta^{mTSP} C_{\max}^* - (6\delta^{mTSP} - \alpha_1)\tau_e$. As $6\delta^{mTSP} - \alpha_1 \geq 0$, $C_{\max} \leq 6\delta^{mTSP} C_{\max}^*$.

If $\alpha_1 > 6\delta^{mTSP}$, Equation (3.4) gives $C_{\max} \leq \alpha_1 \tau_e + 3\delta^{mTSP} D_m^*$. Since $\alpha_1 \tau_e \leq C_{\max}^*$ and $3\delta^{mTSP} D_m^* \leq 6\delta^{mTSP} C_{\max}^*$ due to Equation (3.5), we get $C_{\max} \leq (1 + 6\delta^{mTSP}) C_{\max}^*$. □

Note that in real life systems, it is unusual to have problems with α_1 larger than $6\delta^{mTSP}$ since such a system would be extremely overloaded. Note also that if we use the 2-approximation algorithm of Frederickson et al. [1978] for Minmax m-TSP and set $\delta^{mTSP} = 2$, we obtain a 13-approximation algorithm. Hence, mCfPP($C_{\max}$) admits a constant-factor approximation algorithm.

Since we have an approximation preserving reduction in the proof of Lemma 3.2.2, we also note that the approximation ratio in Theorem 3.4.1 is also valid for the scheduling problem of Section 3.2.1. From Lemma 3.2.2 and Theorem 3.4.1, we deduce the following corollary as a side result.

Corollary 3.4.1. *The scheduling problem,* $FH2, (Pm^{(1)}, 1^{(2)}), (\beta \rightarrow \gamma)||C_{\max}$, *is* $(6\delta^{mTSP} + 1)$*-approximable, where* δ^{mTSP} *is the approximation ratio for Minmax m-TSP.*

The approximation ratio of $(6\delta^{mTSP} + c)$ where $c \in \{0, 1\}$ can be strengthened through an assumption of $\tau_e \geq D_m^*$. Note that this assumption is realistic since it requires that all sites can be visited at least once by m vehicles before τ_e. Then, for $\alpha_1 < 1$, $C_{\max} \leq \tau_e + 3\delta^{mTSP} D_m^* \leq \tau_e(1 + 3\delta^{mTSP})$. Since $C^*_{\max} \geq \tau_e$ due to Equation(5), $C_{\max} \leq (1 + 3\delta^{mTSP})C^*_{\max}$.

For $\alpha_1 \geq 1$, $C_{\max} \leq \alpha_1\tau_e + 3\delta^{mTSP} D_m^* \leq \tau_e(\alpha_1 + 3\delta^{mTSP})$. Since $C^*_{\max} \geq \alpha_1\tau_e$ due to Equation(5), $C_{\max} \leq (1 + 3\delta^{mTSP}/\alpha_1)C^*_{\max}$.

Therefore, $C_{\max} \leq C^*_{\max}(1 + 3\delta^{mTSP})/\max\{1, \alpha_1\}$.

Corollary 3.4.2. *When $\tau_e \geq D_m^*$, our clustering-based algorithm is a $(1+3\delta^{mTSP}/\max\{1, \alpha_1\})$-approximation algorithm for mCfPP($C_{\max}$), where δ^{mTSP} is the approximation ratio for Minmax m-TSP.*

If the 2-approximation algorithm of Frederickson et al. [1978] is used for Minmax m-TSP, then, in case of $\tau_e \geq D_m^*$, the approximation ratio of our algorithm reduces to 7 when $\alpha_1 \leq 1$, and gets even smaller when $\alpha_1 > 1$.

3.5 Conclusions and summary of contributions

In this chapter, we define the collection for processing problem with multiple vehicles and the makespan objective (mCfPP($C_{\max}$)). We prove that the problem is NP-hard by a reduction from a two-stage, hybrid flowshop scheduling problem. We analyze the special case with a single site to find the number of vehicles necessary to achieve the minimum makespan and identify the minimum makespan with a single vehicle. Using the insights obtained from these results, we develop a clustering-based heuristic. We provide the first approximation result for mCfPP($C_{\max}$) and show that a constant-factor approximation exists for this strongly NP-hard problem.

Chapter 4

SINGLE VEHICLE PROBLEM WITH TWO PRIORITIZED OBJECTIVES (CFPP)

4.1 Introduction

This chapter studies the single vehicle collection for processing problem, in which the items that accumulate at a number of sites during $(0, \tau_e]$ should be transported by a single uncapacitated vehicle to the processing facility, which has a limited processing capacity. As in the multiple vehicles case, there are a number of alternative objectives in the single vehicle CfPP such as minimizing maximum completion time of processing of all items (i.e. makespan), minimizing total idle time of the processor, or maximizing the processed amount by a deadline. In this chapter, we study the CfPP with two hierarchical objectives. The first level objective is to maximize the processed amount by a deadline, such as the end time of processing at the processing facility, whereas the second level objective is the minimization of transportation costs, which are assumed to be directly proportional to the distance traveled by the vehicle. In particular, in this chapter, we seek a solution that minimizes the transportation costs while maintaining the maximized total processed amount obtained by optimizing only with respect to the first level objective.

The major difference of this problem from the vehicle routing problems with cost objectives is that the processed amount by a deadline is considered as a prioritized objective, in addition to the transportation cost, since collection strategies that only minimize transportation cost may result in excessive idle times at the processing facility. On the other hand, keeping the processing unit busy requires more frequent, and possibly more costly trips. Furthermore, some distant customers may accumulate specimens at a higher rate, while some close ones may have light patient traffic. The choice of which customer to collect from at what time further complicates the problem.

In this chapter, we prove that the CfPP with two hierarchical objectives is NP-hard, and introduce a Mixed Integer Programming (MIP) formulation for the problem. We characterize properties of optimal solutions and provide methods to bound the two objectives. For the challenging task of generating strong lower bounds on the transportation cost for a given processed amount, we derive valid inequalities and develop two relaxed MIP models that rely on these inequalities and an alternative flow formulation.

The remainder of this chapter is organized as follows. Section 4.2 overviews the previous work on multi-objective optimization and relevant heuristic approaches. In Section 4.3, we provide the problem definition and some propositions. Section 4.4 derives an upper bound on the processed amount by a deadline. In Section 4.5, we provide an MIP formulation and describe two relaxations of the MIP to generate lower bounds on the transportation cost for a given processed amount. Section 4.6 concludes with a summary and contributions of this chapter.

4.2 Previous work on multi-objective optimization and relevant heuristic approaches

In Chapter 2, we provide the previous work on the routing problems that possess similar characteristics with the collection for processing problem. In this section, we focus on the related literature on the single vehicle collection for processing problem with two hierarchical objectives. Since the problem falls into the more general field of multi-objective optimization, also known as multi-criteria decision making, we first provide the previous work on multi-criteria decision making in Section 4.2.1. Next, Section 4.2.2 overviews the possible heuristic approaches to solve the CfPP.

4.2.1 Studies on multi-objective combinatorial optimization problems

In recent years, there has been growing interest in multi-objective optimization due to their potential application to real world problems. In multi-objective optimization problems (MOOPs), a solution is said to *dominate* another only if it has superior performance in all criteria. Thus, MOOP, first, aims to identify *non-dominated* solutions by means of optimization algorithms, then, selects the final solution alternative from a set of non-dominated solutions depending on the information coming from the so-called decision maker. Depending on the timing of this information, three approaches to multi-criteria decision making can be classified: a priori, interactive, and a posteriori decision making.

In CfPP, the objectives have a priori priorities. For the corresponding class of MOOPs, there are a number of alternative approaches in the literature. One commonly used approach [e.g. Kim and de Weck, 2005] is generating a composite single objective function from the multiple objectives, generally, by taking a weighted sum of the objectives. The main drawback of this approach is that a search algorithm based on this approach might not identify a non-dominated solution as its composite single objective value might be worse than that of another solution. Another approach is the ϵ-constraint method [e.g. Haimes et al., 1971], where the decision maker gives constraints for all objectives except the one that remains to be optimized. However, in this approach the solution to the problem largely depends on the selection of the ϵ vector. Goal programming, where the decision maker gives a goal vector of desired objective values and the distance to this vector is to be minimized, is another approach for the MOOPs. However, without a priori knowledge, the goal vector may be unattainable or trivial [Streichert and Tanaka-Yamawaki, 2006]. An emerging approach in multi-objective optimization with a priori priorities is the lexicographic approach. In this approach, the objective with the first priority is optimized first, then, among all alternative optimal solutions the objective with the second priority is optimized next, and so on, until the final objective is optimized [e.g. Weber et al., 2002]. In this thesis, we provide both a lexicographic approach (in Chapter 5) and a novel approach (in Chapter 6) to address our problem that have hierarchical objectives. The novel approach provided in Chapter 6 embeds bicriteria decision making inside a single step tabu search framework by means of classification of the neighborhood solutions according to a dominance relation at each iteration. We note that the adaptation of metaheuristic techniques for the solution of the general field of MOOPs has increased over the last years, giving birth to multi-objective metaheuristics [Ehrgott and Gandibleux, 2004]. For example, there are successful applications of tabu search and hybrid tabu search-based methods on MOOPs, in the literature. As examples, Ben Abdelaziz et al. [1999] presents a multi-objective hybrid heuristic, which is a mix of a tabu search and a genetic algorithm, for the knapsack problem, Gandibleux and Freville [2000] solves 0-1 knapsack problems with multiple linear objectives by a tabu search approach, and Caballero et al. [2007] presents a multi-objective location routing problem and solves it by a multi-objective tabu search procedure. For extensive review of metaheuristics designed for MOOPs, see Ehrgott and Gandibleux [2004].

4.2.2 Possible heuristic approaches for CfPP

The CfPP can be considered as a variant of vehicle routing problem (VRP) with additional features. A general solution approach used in many VRP variants with time considerations is performing routing first and then scheduling the resulting routes, see e.g., Taillard et al. [1996] for VRP with multiple tours, Russell and Gribbin [1991] for periodic VRP, and Koskosidis et al. [1992], Ibaraki et al. [2005] for VRP with time windows. This general framework cannot be used in the case of CfPP since routes should be determined according to the time-varying queue sizes at the processing facility as well as at sites. Routing and scheduling without considering the processing rate and the accumulated amounts might lead to idle times in the processing unit, which may result in unprocessed items by given deadline.

In the literature, tabu search algorithms has been used effectively to solve variants of VRP with single objective, e.g., VRP with time windows [Taillard et al., 1997, Chiang and Russell, 1997, Cordeau et al., 2001], VRP with split delivery [Archetti et al., 2006], the pick up and delivery problem [Bianchessi and Righini, 2006], and the stochastic VRP [Gendreau et al., 1996]. It has been shown that tabu search generally yields very good results on a set of benchmark problems and some larger instances [Gendreau et al., 2002]. Tang and Miller-Hooks [2005] applied tabu search to the team orienteering problem (TOP) and Tang et al. [2007] utilized a tabu search-based approach to solve a variant of TOP with time-dependent rewards. Both of these tabu search applications provide near optimal solutions for actual large-size problems within reasonable computation time. However, the complexity of the problem we studied, which has both routing and scheduling aspects and multiple prioritized objectives, requires a more sophisticated heuristic search of the solution space to get high quality solutions.

For solving complex real world problems, using matheuristics that incorporate MIP solvers or customized MIP codes in a heuristic context, both as primary solvers or as subprocedures, is an emerging field. In this thesis, in Chapter 6 we propose a bicriteria matheuristic that combines a tabu search scheme with an exact linear programming (LP) method to solve the CfPP. The effectiveness of the heuristic is proved over a set of benchmark instances. The major difference of the proposed matheuristic from a classical tabu search application is due to the multiple hierarchical objectives and the hybridization method.

4.3 Problem description

The CfPP is formally defined as follows. We assume that items accumulate between time 0 and time τ_e, and processing can be performed at the processing facility between time 0 and time τ_f. The problem is defined on a directed graph $G = (N^+, A)$, where $N^+ = \{0, 1, 2, \ldots, n\}$ is the node set and $A = \{(i, j) \mid i, j \in N^+\}$ is the arc set. Each arc (i, j) is associated with non-negative travel time t_{ij} and distance d_{ij}. We assume that both the distances and the travel times are metric. Node 0 represents the processing facility, and $N = N^+ \setminus \{0\}$ is the set of sites. Items accumulate at each site $i \in N$ at a constant and known rate of λ_i units per unit time. Without loss of generality, we label the sites so that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Items are processed at the processing facility at a constant and known rate of μ units per unit time. $Q(t)$ represents the amount of items waiting in the queue or undergoing processing at the processing facility at time t. We assume $Q(0) = 0$. Note that, in the clinical testing context, the items represent the specimens, τ_e stands for the end of working day at sites, and τ_f corresponds to the time of day that the processing ends at the processing facility.

The CfPP considers the decisions regarding a single vehicle dedicated to serve the node set, N^+. The vehicle is assumed to have sufficiently large capacity to transfer the total accumulation of all sites, which is a realistic assumption for clinical laboratory logistics. We assume that the vehicle is positioned at the processing facility at time 0.

The set of feasible scheduled giant-tours for a CfPP instance is denoted by $\mathcal{G}$. For a solution $g \in \mathcal{G}$, $P^g(\tau_f)$ represents the processed amount by time τ_f and $C^g(\tau_f)$ represents the transportation cost, which is directly proportional with the distance traveled. The objective is to maximize the processed amount by time τ_f , first, over the set $\mathcal{G}$, and then to select a feasible solution with minimum transportation cost among the ones with maximum processed amount. Such solutions are called *optimal*.

For a CfPP instance, the total processing capacity of the processing unit, i.e., the maximum amount of items that can be processed between times 0 and τ_f, is equal to $\tau_f \mu$. The total accumulated amount, i.e., the amount of items accumulated at all sites between time 0 and time τ_e, is equal to $\tau_e \sum_{i \in N} \lambda_i$, all of which must be transported to the processing facility.

Definition 4.3.1. *The* workload level*,* α_2*, for a CfPP instance is defined as the ratio of the total accumulated amount to the processing capacity. That is,*

$$\alpha_2 = \frac{\tau_e \sum\limits_{i \in N} \lambda_i}{\tau_f \mu}. \tag{4.1}$$

Note that, $\alpha_2 = \alpha_1 \tau_e / \tau_f$.

In systems with heavy workload (i.e., large α_2), it may not be possible to process all of the accumulated amount by τ_f. On the other hand, when the workload is relatively low (i.e., small α_2), there may be many solutions that can process all of the accumulated amount on time, in which case minimization of the transportation cost gains importance.

Let us consider a small illustrative example provided in Figure 4.1. Suppose that $\tau_e = 540$ and $\tau_f = 1200$ minutes, the processing rate is $\mu = 100$ items per hour and there are four sites with accumulation rates of $\lambda_1 = 50$, $\lambda_2 = 106$, $\lambda_3 = 30$, and $\lambda_4 = 7$ items per hour. Symmetric distances are given as $d_{01} = 15, d_{02} = 25, d_{03} = 24, d_{04} = 36, d_{12} = 11, d_{13} = 18, d_{14} = 22, d_{23} = 10, d_{24} = 15$, and $d_{34} = 15$ kilometers. For simplicity, we take $t_{ij} = d_{ij}$ assuming that the speed of the vehicle is one kilometer per minute.

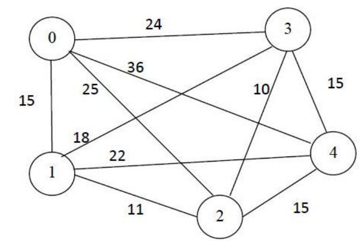


Figure 4.1: A CfPP instance

In this CfPP instance, the total accumulated amount is $\tau_e \sum_{i \in N} \lambda_i = 1737$, whereas the processing capacity is $\tau_f \mu = 2000$. Therefore, $\alpha_2 = 1737/2000 = 0.869$. This workload level indicates that it may be possible to process all items depending on the transportation decisions.

A solution for a CfPP instance has two components. The first component is the sequence of nodes to be visited between 0 and τ_f. This component is also referred to as a giant-tour, where a *giant-tour* is a series of tours, each of which starts and ends at the processing facility, and visits a subset of sites in N. Since the vehicle can wait at a node, the second component of a CfPP solution provides the waiting times at the visited nodes. Hence, the travel times between nodes and waiting times at the nodes determine the node visit times. Accordingly, a CfPP solution is referred to as a *scheduled giant-tour*. The number of tours that the vehicle can perform between times 0 and τ_f is, naturally, bounded. Accordingly, we assume that the vehicle performs at most κ_1 tours during $[0, \tau_e)$ and at most κ_2 tours during $[\tau_e, \tau_f]$. The former ones are referred to as *D-tours* since they start during the day time and the latter ones are referred to as *A-tours* to indicate that they start after the day time office hours. In order for a solution to be *feasible*, it should visit all sites after time τ_e to collect all the remaining accumulated items, and should be completed by time τ_f. Note that it is sufficient to visit a site just once after τ_e, since accumulation at the sites stop at this time. Since there are a total of n sites, the vehicle performs at most n tours after time τ_e, i.e., $1 \leq \kappa_2 \leq n$. We note that the vehicle is allowed to be touring at time τ_e. Therefore, the last D-tour of a scheduled giant-tour might visit a number of sites after τ_e. Figure 4.2 illustrates a general time line representation of a scheduled giant-tour as well as the queue size at the processing facility in the time line.

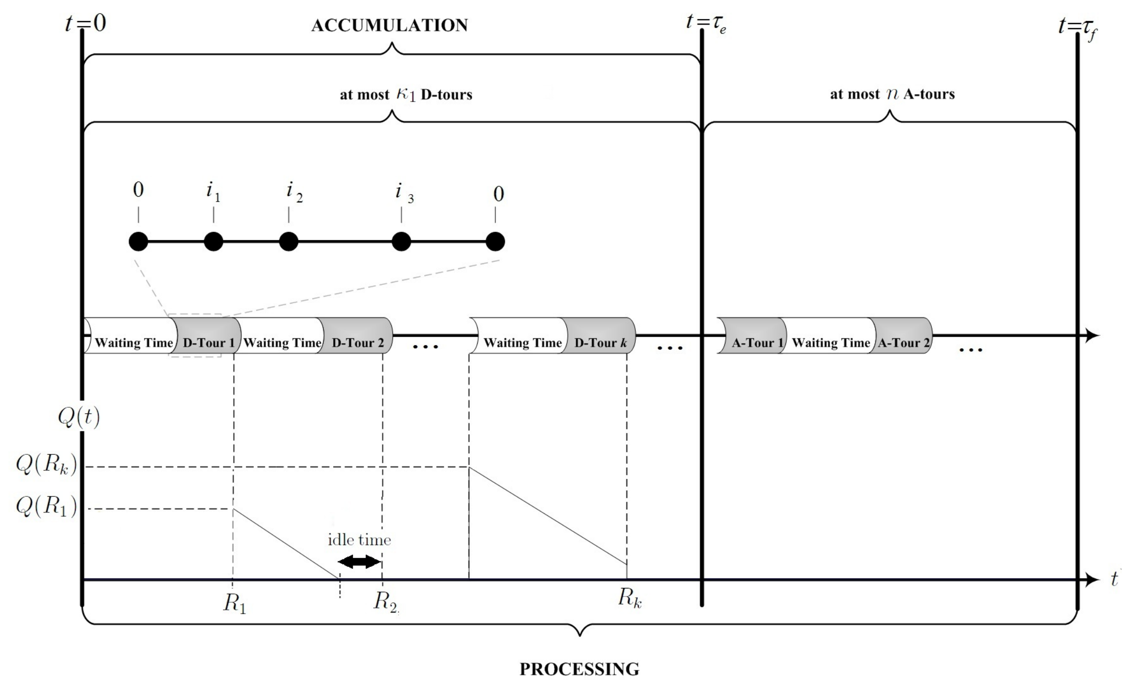


Figure 4.2: Graphical time representation for a CfPP solution

For our previous example, if the tours performed by the vehicle throughout the day are 0-1-2-4-0, 0-2-3-0, 0-1-0, and 0-2-4-3-0, then the corresponding giant-tour is 0-1-2-4-0-2-3-0-1-0-2-4-3-0. There might be more than one feasible solution, i.e., scheduled giant-tour, for this sequence of nodes. One such solution is 0-1-2-4-0(153)-2-3-0(251)-1-0-2-4-3-0, where the numbers in parentheses denote the waiting times in minutes at the corresponding nodes. The vehicle waits only at the processing facility in this solution.

An optimal solution for our example is 0(74.7)-2-1-0(142.1)-2-1-0(221.2)-1-2-4-3-0, with $P(\tau_f) = 1737$ and $C(\tau_f) = 182$. In this solution, all the accumulated amount is processed. The graphical time representation of the solution and the amount of unprocessed items at the processing facility (i.e., $Q(t)$) in time interval $[0, \tau_f]$ are illustrated in Figure 4.3. We see that the processor remains idle during the first tour and for a short interval during the second tour.

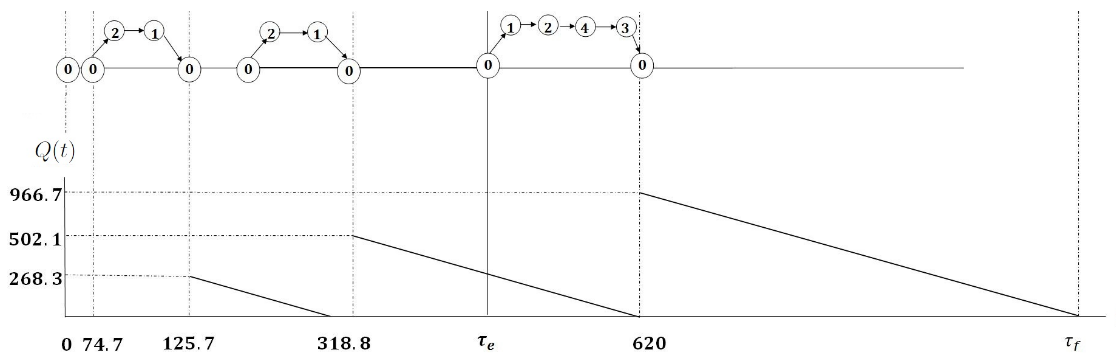


Figure 4.3: The graphical time representation of an optimal solution for the example

Keeping the processing unit utilized by replenishing its queue of units to be processed serves to maximize the processed amount by time τ_f. Frequent deliveries as well as deliveries with large specimen amounts increase the utilization of the processing unit. However, visiting an additional site increases both the collected specimen amount and the tour duration. Hence, having tours with long durations, which might cause idle time at the processing unit, becomes critical when the work load is heavy, although it can be tolerated in lightly loaded systems.

We next investigate the problem difficulty and certain properties of optimal solutions.

4.3.1 NP-hardness of CfPP

Proposition 4.3.1. *CfPP is NP-hard.*

Proof. The proof is by reduction from TSP, which is shown to be NP-hard [Golden et al., 1987]. In TSP, a graph $G = (N^+, E)$, where $N^+ = \{0, 1, \dots, n\}$ denotes the set of cities and E denotes the set of edges, (i, j), with associated travel times, d_{ij}, is given. The problem is to find the cheapest path visiting all of the cities and returning to the starting point. Let the traveling time of the optimal TSP solution be L.

Given an instance of TSP, we define an instance of CfPP on the same graph $G = (N^+, E)$, where $N = N^+ \setminus \{0\}$ denotes the set of n sites and node 0 denotes the processing facility with processing rate μ and E denotes the set of edges, (i, j), with associated travel times and distances equal to d_{ij}. Set $\tau_e = \min_{i \in N} t_{0i}$ so that the accumulation will have ended at the earliest visiting time of any site. Set total accumulation to μL. Assign each site $i \in N$ an accumulation rate of $(\mu L)/(n\tau_e))$. Set $\tau_f = 2L$. Clearly, the construction of CfPP instance takes polynomial time.

Suppose that there exists an optimal TSP solution with traveling time L. We construct a CfPP solution with a single tour that starts at time 0, follows the optimal TSP tour, and ends at time L. At the end of the tour, μL units will be available at the processing facility. The processing ends at time $2L$. Note that this is an optimal solution for the corresponding CfPP instance, because all of the accumulated items are processed and, furthermore, this can be achieved at minimum transportation cost since the TSP tour provides the minimum travel cost visiting every site.

Now let us consider an optimal CfPP solution, g^*, with a total processed amount of $P^*(\tau_f)$. If the solution performs more than one tour, then its transportation cost is strictly greater than L. Now let us consider another CfPP solution, g, with a single tour that starts at time 0, follows the optimal TSP tour, and ends at time L. Clearly, the processed amount for this solution is $P^*(\tau_f)$ since it can process all the accumulated amount. However, g has a smaller transportation cost than that of g^*, which leads to contradiction. Hence, the optimal CfPP solution is optimal for the given TSP instance.

$\square$

4.3.2 Properties of some optimal solutions of a CfPP instance

The set of feasible solutions, $\mathcal{G}$, may be extremely large even for problems with limited number of nodes (e.g., $n \geq 10$). In order to simplify the search for optimal solutions in a large feasible solution space, we investigate the structure of optimal solutions. The following two propositions allow us to concentrate on certain types of solutions.

Proposition 4.3.2. *There exists an optimal solution in which the vehicle does not wait at any site.*

Proof. Assume for contradiction that in all optimal solutions the vehicle waits at some site in some tour. Consider an optimal solution, g^*, which dictates a vehicle to start a tour k specified by the sequence of nodes $0\, i_1\, i_2\, \cdots\, i_p\, 0$ at time t_{k_0}. In this tour, the vehicle waits for $w > 0$ time units at the site i_j for some $j \in \{1, 2, \ldots, p\}$. Now, let solution g perform the same giant tour as g^*, but it starts tour k at time $t_{k_0} + w$ and does not let the vehicle wait at site i_j.

Case 1: $k = 1, 2, \ldots, \kappa_1$

The vehicle will arrive at the sites $i_j, i_{j+1}, \cdots, i_p$ at the same time under both solutions, so the amount collected from these sites in tour k are equal for both solutions. However, solution g collects an extra amount of $w \sum_{l=1}^{j-1} \lambda_{i_l}$ items from the sites $i_1, i_2, \ldots, i_{j-1}$ in tour k, since a vehicle following solution g arrives to each of these sites w time units later than a vehicle following g^*. As all the accumulation rates λ_i and w are strictly positive, solution g collects more than solution g^* in tour k. In the tours before tour k, both solutions collect the same amount of items, since the node visit times are the same. In the tours after tour k, which visits any site i_l where $l \in \{1, 2, \cdots, j-1\}$ at the first time after tour k, solution g collects less than solution g^* from site i_l, where the difference is $w\lambda_{i_l}$. Since the return time to the processing facility for each tour is the same for both solutions, in solution g, $w\sum_{l=1}^{j-1}\lambda_{i_l}$ items arrive to the processing facility earlier. Therefore, solution g is at least as good as solution g^*, which is a contradiction.

Case 2: $k = \kappa_1 + 1, \ldots, \kappa_1 + \kappa_2$

As tour k starts when there is no accumulation, the collected amounts in tour k and in the tours after tour k are equal in solutions g and g^*. In the tours before tour k, both solutions collect the same amount, since the node visit times are the same. Therefore, similar to Case 1, in solution g, $\delta\sum_{l=1}^{m-1}\lambda_{i_l}$ items arrive to the processing facility earlier.Thus, solution g is at least as good as solution g^*, which is a contradiction. $\square$

Proposition 4.3.3. *There exists an optimal solution in which sites are not visited more than once in any tour.*

Proof. Assume for contradiction that in all optimal solutions there exits some site that is visited more than once in some tour. Consider an optimal solution, g^*, in which tour k starts at time t_{k_0} and follows the sequence $0\, i_1 \,\cdots\, i_m \,\cdots\, i_{m'} \,\cdots\, i_p\, 0$, where $i_m = i_{m'} = j$. Now let solution g start tour k at time $t_{k_0} + \delta$, where $\delta = d_{i_{m-1}i_m} + d_{i_m i_{m+1}} - d_{i_{m-1}i_{m+1}}$ to follow the route $0\, i_1 \,\cdots\, i_{m-1}\, i_{m+1} \cdots (i_{m'} = j) \cdots i_p\, 0$. Note that, $\delta > 0$ due to the assumed triangle inequality for time and distance.

Case 1: $k = 1, 2, \ldots, \kappa_1$

The vehicle will arrive at the nodes $i_{m+1}, i_{m+2}, \ldots, i_p$ at the same time under both solutions, so the total amount collected from these nodes in tour k are equal for both solutions. Note that, the total amount collected from site j in tour k under solution g^* and solution g are also the same, since the last time that the vehicle visits site

j is the same under both solutions. However, solution g collects an extra amount of $\delta \sum_{l=1}^{m-1} \lambda_{i_l}$ items from the nodes $i_1, i_2 \cdots i_{m-1}$ in tour k, since a vehicle following solution g arrives to each of these nodes δ time units later than a vehicle following g^*. As all the accumulation rates λ_i and δ are strictly positive, solution g collects more than solution g^* in tour k. In the tours after tour k, which visits any site i_l where $l \in \{1, 2, \cdots, m-1\}$ at the first time after tour k, solution g collects less than solution g^* from site i_l, where the difference is $\delta\lambda_{i_l}$. Since the return time to the processing facility for each tour is the same for both solutions, in solution g, $\delta \sum_{l=1}^{m-1} \lambda_{i_l}$ items arrive to the processing facility earlier.Therefore, solution g is at least as good as solution g^*, which is a contradiction.

Case 2: $k = \kappa_1 + 1, \ldots, \kappa_1 + \kappa_2$

As tour k starts when there is no accumulation, the collected amounts in tour k and in the tours after tour k are equal in solutions g and g^*. In the tours before tour k, both solutions collect the same amount of items, since the site visit times are the same. Therefore, similar to Case 1, in solution g, $\delta \sum_{l=1}^{m-1} \lambda_{i_l}$ items arrive to the processing facility earlier.Thus, solution g is at least as good as solution g^*, which is a contradiction. □

Since any solution that does not satisfy these properties can be converted to one with these properties without any loss of performance, we search only for such solutions.

4.4 An upper bound on the processed amount by a deadline

We next provide a strong upper bound for the processed amount by a deadline. We use this upper bound to analyze the performance of our solution approaches with respect to the first level objective of maximizing the processed amount by time τ_f.

The processed amount by time τ_f, $P(\tau_f)$, cannot exceed the total accumulated amount or the total processing capacity. Therefore, $\min\{\tau_e \sum_{i\in N} \lambda_i, \tau_f \mu\}$ provides an upper bound for $P(\tau_f)$. However, this bound may be quite loose, since it does not consider the time required for the transportation of items from the sites to the processing facility. In order to obtain a tighter bound, we relax the resource constraint on the number of vehicles. That is, we assume that infinitely many vehicles are available.

It takes at least t_{i0} time units to transfer any item from site i to the processing facility in a tour. If there is no limit on the number of vehicles available at the processing facility, a vehicle can be dispatched to each site every minute. Then, the first delivery of items to the processing facility occurs at time $\min_{i\in N}\{t_{0i} + t_{i0}\}$. If $Q(0) = 0$, then the processing facility is idle until time $\min_{i\in N}\{t_{0i} + t_{i0}\}$. The accumulation of items at the sites ends at time τ_e. Therefore, the latest arrival of items to the processing facility will occur at time $t = \tau_e + \max_{i\in N}\{t_{i0}\}$ from the farthest site. We define $\mathrm{UB}_{P(\tau_f)}$ as the processed amount until τ_f, and calculate $\mathrm{UB}_{P(\tau_f)}$ using the pseudocode provided in Algorithm 1.

Let us calculate $\mathrm{UB}_{P(\tau_f)}$ for the previously discussed example where $\alpha_2 = 0.869$. We also analyze the same system for α_2 values of 1 and 1.2 by changing μ accordingly. Figure 4.4 illustrates the amount of unprocessed items in the processing facility during the time interval $[0, \tau_f]$ for $\alpha_2 = 0.869, 1$ and $\alpha_2 = 1.2$. When $\alpha_2 = 0.869$, $Q(\tau_f) = 0$, meaning that all the accumulated amount can be processed, that is $\mathrm{UB}_{P(\tau_f)} = \tau_e \sum_{i\in N} \lambda_i$. When $\alpha_2 \geq 1$, however, we have $Q(\tau_f) = 45$, meaning that the accumulated amount cannot be processed completely. When $\alpha_2 = 1.2$, the amount of unprocessed items by time τ_f increases to 327.

Algorithm 1 $\text{UB}_{P(\tau_f)}$ Algorithm

Input: N^+, t_{ij}, λ_i

Output: $\text{UB}_{P(\tau_f)}$

Set $Q(0), P(0) \leftarrow 0$.
for $t \in \{1, 2, \ldots, \tau_f\}$ **do**
    Set $\gamma \leftarrow 0$. // γ denotes the amount of items that arrive at time t.
    for $i \in N$ **do**
        if $t = t_{0i} + t_{i0}$ **then**
            Set $\gamma \leftarrow \gamma + t_{0i} * \lambda_i$. // The first item arrival from node i.
        else if $t_{0i} + t_{i0} < t < \tau_e + t_{i0}$ **then**
            Set $\gamma \leftarrow \gamma + \lambda_i$.
        else
            // Do nothing: no item arrival from node i.
        end if
    end for
    Set $P(t) \leftarrow P(t-1) + \min\{Q(t-1), \mu\}$. // Calculate the amount of processed items during time t.
    Set $Q(t) \leftarrow Q(t-1) + \gamma - \min\{Q(t-1), \mu\}$. // Update the amount of unprocessed items in the processing facility.
end for
Set $\text{UB}_{P(\tau_f)} \leftarrow P(\tau_f)$.

4.5 *The mathematical model*

In this section, we develop a mixed integer programming model that determines the number of tours to be performed by the vehicle until time τ_f, as well as the start

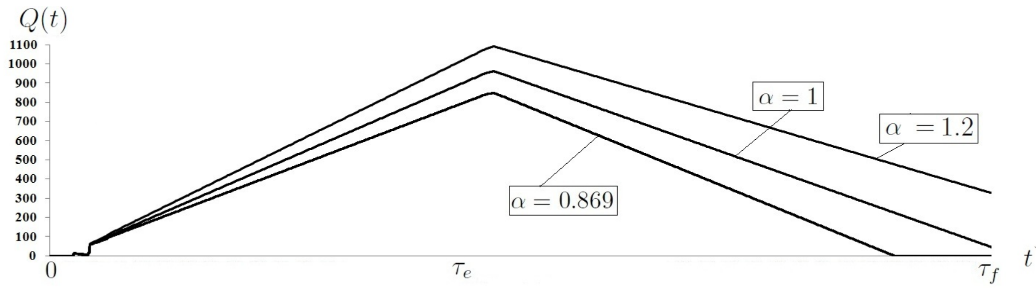


Figure 4.4: The amount of unprocessed items in the processing facility for the example when there is no limit on the number of vehicles

and end times of the tours throughout the planning horizon. It also selects the sites to be visited and their sequence in each tour. We do not have a fixed number of tours. Hence, the model puts an upper bound on the tour number and allows empty tours. To calculate the accumulated amount at each node, we keep track of the last visit time of each node. Moreover, we need to know whether a node is visited before time τ_e or not since accumulation stops at τ_e. The model ensures that all accumulated items are collected by the vehicle. The model allows the vehicle to wait only at the processing facility before τ_e due to Proposition 4.3.2, and does not allow visiting a site more than once in a tour due to Proposition 4.3.3. We define the decision variables and constraints as follows.

Decision variables

X_{ijk} binary variable indicating if node j is visited immediately after node i in tour k $((i,j) \in A,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2)$. If tour k is an empty tour, then $X_{00k} = 1$ and $X_{0ik} = 0,\ X_{i0k} = 0,\ X_{ijk} = 0,\ \forall i, j \in N$.

Y_{ik} binary variable indicating if node i is visited in tour k $(i \in N,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2)$.

T_{ik} visit time of node i at tour k ($i \in N^+, k = 1, 2, \ldots, \kappa_1 + \kappa_2$). If node i is not visited in tour k, it denotes the last visit time of node i before tour k (i.e., $T_{ik} = T_{i,k-1}$). T_{0k} denotes the starting time of tour k.

R_k return time of the vehicle from tour k to processing facility ($k = 1, \ldots, \kappa_1+\kappa_2$); $R_0 = 0$.

E_{ik} collected amount from site i in tour k ($i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

S_k total collected amount in tour k ($k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

L_{ik} auxiliary variable used to calculate E_{ik} ($i \in N$, $k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

W_k waiting time of the vehicle at the processing facility at the beginning of tour k ($k = 1, 2, \ldots, \kappa_1 + \kappa_2$) .

Q_k amount of unprocessed items at the processing facility at the end of tour k ($k = 0, 1, 2, \ldots, \kappa_1 + \kappa_2$). Q_0 is a parameter which represents the number of unprocessed items at the beginning of the day.

U_k amount of processed items between R_k and R_{k+1}($k = 0, 1, 2, \ldots, \kappa_1 + \kappa_2 - 1$).

$U_{\kappa_1+\kappa_2}$ amount of processed items between $R_{\kappa_1+\kappa_2}$ and τ_f.

Constraints

$$\sum_{j \in N^+} X_{0jk} = 1, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.2a}$$

$$\sum_{j \in N^+} X_{j0k} = 1, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.2b}$$

$$\sum_{j \in N^+, j \neq i} X_{ijk} = Y_{ik}, \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.2c}$$

$$\sum_{j \in N^+, j \neq i} X_{ijk} - \sum_{j \in N^+, j \neq i} X_{jik} = 0, \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.2d}$$

$$R_k = R_{k-1} + W_k + \sum_{(i,j)\in A} t_{ij} X_{ijk}, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.3a}$$

$$T_{0k} = R_{k-1} + W_k, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.3b}$$

$$R_{\kappa_1+\kappa_2} \le \tau_f, \tag{4.3c}$$

$$T_{0,\kappa_1} \le \tau_e, \tag{4.3d}$$

$$T_{0,\kappa_1+1} \ge \tau_e, \tag{4.3e}$$

$$T_{ik} + t_{ij} - T_{jk} \le M(1 - X_{ijk}), \qquad \forall (i,j) \in A, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.4a}$$

$$T_{ik} + t_{ij} - T_{jk} \ge -M(1 - X_{ijk}), \qquad \forall (i,j) \in A, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.4b}$$

$$T_{i,k+1} - T_{ik} \le MY_{i,k+1}, \qquad \forall i \in N^+, k = 1, 2, \ldots, \kappa_1 + \kappa_2 - 1, \tag{4.4c}$$

$$T_{i,k+1} \ge T_{ik}, \qquad \forall i \in N^+, k = 1, 2, \ldots, \kappa_1 + \kappa_2 - 1, \tag{4.4d}$$

$$L_{ik} \le T_{ik}, \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.5a}$$

$$L_{ik} \le \tau_e, \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.5b}$$

$$E_{ik} = \lambda_i (L_{ik} - L_{i,k-1}), \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.5c}$$

$$L_{i0} = 0, \qquad \forall i \in N, \tag{4.5d}$$

$$S_k = \sum_{i\in N} E_{ik}, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.5e}$$

$$\sum_{k=1}^{\kappa_1+\kappa_2} E_{ik} = \lambda_i \tau_e, \qquad \forall i \in N, \tag{4.5f}$$

$$U_k \le \mu(R_{k+1} - R_k), \qquad \forall k = 0, 1, \ldots, \kappa_1 + \kappa_2 - 1, \tag{4.6a}$$

$$U_{\kappa_1+\kappa_2} \le \mu(\tau_f - R_{\kappa_1+\kappa_2}), \tag{4.6b}$$

$$U_k \le Q_k, \qquad \forall k = 0, 1, \ldots, \kappa_1 + \kappa_2, \tag{4.6c}$$

$$Q_k = Q_{k-1} - U_{k-1} + S_k, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.7}$$

$$X_{ijk}, Y_{ik} \in \{0, 1\}, \qquad \forall i, j \in N^+,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.8}$$

$$T_{ik}, R_k, E_{ik}, S_k, L_{ik}, W_k, Q_k, U_k \ge 0, \qquad \forall i \in N^+,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2. \tag{4.9}$$

Using these constraints, we define two models. Since the first level objective is to maximize the processed amount by time τ_f, model M_1 is solved first:

$$\begin{aligned} M_1 : \text{ Maximize } \quad & P(\tau_f) = \sum_{k=1}^{\kappa_1+\kappa_2} U_k \\ \text{subject to } \quad & \text{Constraints } (4.2a) - (4.9). \end{aligned}$$

The solution of M_1 provides the optimal value of $P(\tau_f)$, denoted by $P^*(\tau_f)$, which is then used as a lower bound for the processed amount in model M_2. Hence, model M_2 minimizes the transportation cost subject to this additional constraint:

$$\begin{aligned} M_2 : \text{ Minimize } \quad & C(\tau_f) = \sum_{k=1}^{\kappa_1+\kappa_2} \sum_{(i,j)\in A} d_{ij} X_{ijk} \\ \text{subject to } \quad & \text{Constraints } (4.2a) - (4.9) \\ & \sum_{k=1}^{\kappa_1+\kappa_2} U_k \geq P^*(\tau_f). \end{aligned} \tag{4.10}$$

Constraints (4.2a) and (4.2b) require that each tour begins and ends at the processing facility. The outgoing degree of a node i must be equal to Y_{ik} through Constraint (4.2c). Constraint (4.2d) implies that the flow is balanced at each node.

Constraints (4.3a)-(4.3e) define the relation between tour return times, waiting times, and tour start times. Constraints (4.3a) and (4.3b) calculate the return time and the starting time of a tour, respectively. Constraint (4.3c) guarantees that the last tour should be completed by τ_f and Constraints (4.3d)-(4.3e) limit the number of tours before τ_e by κ_1.

Constraints (4.4a)-(4.4d) define the node visit times. If the vehicle visits node j immediately after node i in tour k, Constraints (4.4a) and (4.4b) together restrict the vehicle to be at node j at time $T_{ik} + t_{ij}$. These constraints also eliminate subtours. Constraints (4.4c) and (4.4d) ensure that if node i is not visited in tour k, T_{ik} equals the last visit time before that tour.

Next, we define Constraints (4.5a)-(4.5f) to calculate the collected amounts from the visited nodes. Since the accumulation at the sites ends at time τ_e,

$$\lambda_i(\min\{T_{ik}, \tau_e\} - \min\{T_{i,k-1}, \tau_e\})$$

gives the collected amount from node i in tour k. Note that, if node i is not visited in tour k, then $T_{ik} = T_{i,k-1}$ and hence $\lambda_i(\min\{T_{ik}, \tau_e\} - \min\{T_{i,k-1}, \tau_e\})$ is zero. We linearize this quantity by defining the auxiliary variables, L_{ik}'s, as $\min\{T_{ik}, \tau_e\}$ through Constraints (4.5a)-(4.5b). By this way, we do not exactly calculate the collected amount from a visited node. However, we guarantee that the vehicle cannot collect an amount that is greater than the accumulated amount by the visit time of the node. Constraints (4.5c)-(4.5e) calculate the collected amount in tour k. Constraint (4.5f) ensures that all of the items accumulated at the nodes are collected.

The amount of processed items between the return times of two consecutive tours, U_k, cannot exceed the amount of unprocessed items, Q_k, and processing capacity, $\mu(R_{k+1} - R_k)$. This condition is satisfied via Constraints (4.6a)-(4.6c).

Constraint (4.7) balances the queue size at the processing facility. Constraint (4.8) defines the binary variables in a tour, while Constraint (4.9) dictates nonnegativity of all variables.

Our computational experiments showed that both M_1 and M_2 can be solved to optimality only for problems with low workload levels (e.g., $\alpha_2 \leq 0.8$) and limited number of nodes (e.g., $n \leq 10$), but verifying optimality takes extensive computation time. In particular, M_2 is significantly more difficult than M_1.

4.5.1 Valid inequalities

The following valid inequalities are used to strengthen the MIP models M_1 and M_2. The formulation with the valid inequalities are computationally tested and the results are provided in Section 5.2.

We first add a valid inequality that relate the binary variables X_{ijk} and Y_{ik} to both of the models, M_1 and M_2. If a node i is not visited in tour k, i.e., $Y_{ik} = 0$, the corresponding routing variables, X_{ijk} and X_{jik}, should also be 0.

$$X_{ijk} \leq Y_{ik}, \quad \forall i \in N, \forall j \in N^+, i \neq j, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.10a}$$

$$X_{jik} \leq Y_{ik}, \quad \forall i \in N, \forall j \in N^+, i \neq j, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.10b}$$

$$X_{ijk} + X_{jik} \leq Y_{ik}, \quad \forall i \in N, \forall j \in N, i \neq j, k = 1, 2, \ldots, \kappa_1 + \kappa_2. \tag{4.10c}$$

Next, to strengthen model M_2, we generate a valid inequality using a lower bound for the processed amount by time τ_f, $P^*(\tau_f)$. In order to process at least $P^*(\tau_f)$ items, the vehicle should complete the first tour by time $\tau_f - P^*(\tau_f)/\mu$. That is, $R_1 \leq \tau_f - P^*(\tau_f)/\mu$. As the vehicle transports S_1 items to the processing facility at the end of tour 1, the processor can process at most S_1 items up to R_2. Therefore, the second tour cannot be completed later than $\tau_f - (P^*(\tau_f) - S_1)/\mu$ so that the remaining $P^*(\tau_f) - S_1$ items can be processed in $(P^*(\tau_f) - S_1)/\mu$ time units. That is, $R_2 \leq \tau_f - P^*(\tau_f)/\mu + S_1/\mu$. Continuing like this, we obtain the following valid inequalities.

Proposition 4.5.1. *The following inequalities are valid for model M_2:*

$$R_k \leq \tau_f - P^*(\tau_f)/\mu + \textstyle\sum_{p=1}^{k-1} S_p/\mu, \quad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2. \tag{4.11}$$

Proof. Assume that for a feasible solution of M_2, for some tour $k = 1, 2, \ldots, \kappa_1 + \kappa_2$, inequality (4.11) is not satisfied, that is, $R_k > \tau_f - P^*(\tau_f)/\mu + \sum_{p=1}^{k-1} S_p/\mu$. Note that during the time interval $[0, R_k]$, $\sum_{p=1}^{k-1} S_p$ items are transferred to the processing facility. Therefore, at most $\sum_{p=1}^{k-1} S_p$ items can be processed in this time interval. Then, the processing capacity during time interval $[R_k, \tau_f]$ is $\mu(\tau_f - R_k) < \mu(\tau_f - (\tau_f - P^*(\tau_f)/\mu + \sum_{p=1}^{k-1} S_p/\mu))$. The right hand side simplifies to $P^*(\tau_f) - \sum_{p=1}^{k-1} S_p$, implying that this solution cannot process $P^*(\tau_f)$ items by time τ_f and violates Constraint (4.10) of M_2, which is a contradiction. □

Proposition 4.5.1 is used in relaxations of model M_2 to generate lower bounds. In addition, we tested the inequalities on a set of instances provided in Section 5.2

and observed that it provides a slight improvement (around 1-2 %) on the best found objective value of M_2 at the end of a six-hour (i.e., 21600 s) run time limit. Hence, we included these inequalities in all of our computational tests.

We performed computational experiments with some of the problem instances provided in Section 5.2.1 on a 3.40 GHz Intel(R) Xeon(TM) processing unit with 8 GB of RAM. GAMS 23.3 with CPLEX solver was used to solve the model. Because of the complexity of the problem, the computational experiments showed that this model can solve problems with only limited size (n=10 with only certain parameter settings). This led us to the heuristic approaches provided in Chapters 5 and 6 to address this problem.

4.5.2 Lower bounds on the transportation cost

In this section, we bound the transportation cost, $C(\tau_f)$, under the condition that at least $P^*(\tau_f)$ items should be processed by time τ_f. We present two relaxations of model M_2. The corresponding MIP formulations are solved with a run time limit and the best found lower bound is kept. The bound is used to evaluate the performance of the heuristic solution approaches provided in Chapters 5 and 6 in terms of the transportation cost.

In model M_2, having Constraint (4.10) narrows the set of feasible solutions dramatically. Especially when the workload level is high, there may be few solutions that achieve $P^*(\tau_f)$. Thus, the branch and bound algorithm has difficulty in finding a candidate feasible solution. To overcome this, we relax Constraint (4.10) and replace it by the valid inequalities given in Constraint (4.11). As a result, the relaxed model cannot guarantee that $P^*(\tau_f)$ items are processed by time τ_f. Another reason for the difficulty of solving M_2 is due to Constraints (4.4a) and (4.4b) that contain big-M. We eliminate Constraints (4.4a) and (4.4b), which calculate the exact values of T_{ik} variables and prevent subtours. Instead, we utilize a single commodity flow formulation and add the following constraints to the MIP:

$$T_{ik} \leq R_k - t_{i0}, \quad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.12}$$

$$\sum_{j \in N^+, j \neq i} G_{ijk} - \sum_{j \in N^+, j \neq i} G_{jik} = \lambda_i (L_{ik} - L_{i,k-1}),\ \forall i \in N,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.13a}$$

$$G_{ijk} \leq X_{ijk} \sum_{i' \in N} \lambda_{i'} \tau_e,\ \forall (i,j) \in E,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.13b}$$

$$G_{ijk} \leq S_k,\ \forall i \in N,\ \forall j \in N^+,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.13c}$$

$$G_{ijk} \geq 0,\ \forall (i,j) \in E,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{4.13d}$$

where G_{ijk} denotes the amount of items carried by the vehicle while traveling from node i to node j in tour k ($i, j \in N^+$ and $k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

Constraint (4.12) puts a bound on node visit time variables T_{ik}. We utilize Constraints (4.13a)-(4.13d) for subtour elimination. For this purpose, single commodity flow variables G_{ijk} are defined akin to the Gavish-Graves formulation for the TSP [Gavish and Graves, 1978]. Constraint (4.13a) balances the flow at each node $i \in N$. Note that these constraints do not calculate the collected amount exactly. The flow between two nodes in a tour can be positive only if the arc between these nodes is traveled in the corresponding tour by Constraint (4.13b), and the amount of the flow is bounded by the amount collected in that tour by Constraint (4.13c). Constraint (4.13d) is the nonnegativity constraint for the flow variables. This MIP constitutes our first relaxation of M_2.

We obtain a second relaxation by excluding the calculations related to the processed amount between consecutive tours from the first relaxation. We eliminate the decision variables U_k and Q_k, as well as the related Constraints (4.6a)-(4.6c) and (4.7).

Our experiments show that despite significant improvements in the progress of the branch and bound algorithm, even these relaxations are difficult to solve to optimality in reasonable time. Hence, we put a time limit for their solution and record the best found lower bound. These bounds bring 74% improvement on the average over the best lower bound obtained by solving M_2 within its run time limit.

4.6 Conclusions and summary of contributions

In this chapter, we define the single vehicle clinical specimen collection problem under the objectives of maximizing the processed amount by a deadline as a first priority and minimizing the transportation costs as a second priority. We show that this problem is NP-hard and provide an MIP formulation for its solution. We present a procedure to calculate an upper bound on the processed amount by a deadline and two relaxed MIP models to generate lower bounds on the transportation cost.

Through a preliminary computational analysis, we conclude that since the solution space is extremely large even for small problem sizes, even the MIP model strengthened with valid inequalities cannot find good solutions within reasonable time. This result leads us to the heuristic solution approaches provided in Chapters 5 and 6.

Chapter 5

AN MIP-BASED HEURISTIC SOLUTION APPROACH FOR THE SINGLE VEHICLE PROBLEM

Like most of the NP-hard problems, it is hard to solve the single vehicle collection for processing problem with two prioritized objectives to optimality for the instances that have realistic-size. In Chapter 4, we observed through preliminary tests that the computation time required to solve the problem to optimality is prohibitive even for small-sized instances. With this motivation, this chapter proposes a heuristic approach that incorporates additional constraints to the MIP model with the goal of reducing the solution space.

In this chapter, we, first, identify rules to eliminate feasible solutions that are likely to be suboptimal. Based on these results, we develop a heuristic approach that solves the MIP model provided in Section 4.5 with additional constraints that reduce the solution space. To evaluate the performance of this approach, we use the bounds on the two objectives provided in Chapter 4. We conduct computational experiments on realistic test instances to demonstrate the effectiveness of our approach. We also extract insights on key problem parameters and their effects on the solutions by further experiments.

The remainder of this chapter is organized as follows. Section 5.1 presents our approach to find a good feasible solution from the MIP formulation. Results from our computational analysis on realistic data are given in Section 5.2. Finally, Section 5.3 concludes and summarizes the main contributions of the chapter.

5.1 Solution approach

The proposed solution approach is based on two approaches that eliminate feasible solutions that are likely to be inferior without significantly compromising from optimality. Both approaches utilize the proposed models in Section 4.5 with some modifications. The first approach is based on restricting the vehicle to perform only one tour after τ_e (Section 5.1.1), whereas the second one identifies, for each tour, a set of sites with low accumulation rates and does not allow the vehicle to visit these sites in the corresponding tour (Section 5.1.2). Although these approaches can be used individually, their joint use has generated better results for specified run time limits in our experiments (Section 5.2.2).

5.1.1 Searching for solutions with a final TSP tour

To cope with the computational difficulty of the proposed models, we first restrict the vehicle to perform only one tour that visits all nodes after τ_e with duration τ and cost θ. Since items do not accumulate after τ_e, the node sequence of the final tour does not affect the collected amount, and so, it is independent of the accumulated amounts at sites. Therefore, routing and scheduling decisions before and after time τ_e become independent. Hence, we first solve a TSP with the objective of minimizing the total traveling time (not distances) to visit all nodes. This constitutes the last tour. We exclude the visiting variables of the last tour from models M_1 and M_2, but in order to calculate the collected and processed amounts we still keep the remaining variables corresponding to the last tour. We generate models M_1^T and M_2^T from M_1 and M_2, correspondingly, according to the following.

- $\kappa_2 = 1$.

- Constraints (4.2a)-(4.2d), (4.3a)-(4.3b), (4.4a)-(4.4b), and (4.8) are defined for

$k = 1, 2, \ldots, \kappa_1$ rather than $k = 1, 2, \ldots, \kappa_1 + \kappa_2$.

- Constraints (4.3c)-(4.3e) are replaced with:

$$R_{\kappa_1+1} = \tau_e + \tau, \tag{5.1a}$$

$$R_{\kappa_1} \leq \tau_e, \tag{5.1b}$$

$$T_{i,\kappa_1+1} = \tau_e, \quad \forall i \in N. \tag{5.1c}$$

- Variables L_{ik}, $\forall i \in N, k = 1, 2, \ldots, \kappa_1 + 1$, are removed so that Constraints (4.5a)-(4.5b) are deleted and Constraints (4.5c) and (4.5d) are replaced with

$$E_{ik} = \lambda_i (T_{ik} - T_{i,k-1}), \quad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + 1, \tag{5.2a}$$

$$T_{i0} = 0, \quad \forall i \in N. \tag{5.2b}$$

After solving M_2^T, the cost of the final TSP tour, θ, is added to the traveling cost.

5.1.2 Node filtering heuristic

In this section, we propose a node filtering heuristic by adding constraints to models M_1 and M_2. These constraints ensure that nodes with low accumulation rates are not allowed to be visited in each tour. By filtering a node, we defer collecting its accumulated amount to a subsequent tour. In order to identify the nodes to be filtered in a tour, we consider the remaining processing capacity and workload. We adjust the remaining processing capacity by assuming that the deferred amount will be processed by τ_f. If this remaining capacity is sufficient to achieve the upper bound on $P(\tau_f)$, we allow filtering this node.

Specifically, to decide on which nodes to filter in tour k, we consider the following quantities. At the return time of tour $k-1$ (i.e., R_{k-1}), the total amount of items processed is $\sum_{r=1}^{k-2} U_r$. Therefore, $\mathrm{UB}_{P(\tau_f)} - \sum_{r=1}^{k-2} U_r$ is an upper bound on the processed amount in the time interval $(R_{k-1}, \tau_f]$, for any $k \leq \kappa_1$. Moreover, the remaining processing capacity is $(\tau_f - R_{k-1})\mu$, while the total accumulation amount in site i is $\lambda_i \tau_e$. Then, the adjusted remaining capacity is $(\tau_f - R_{k-1})\mu - \tau_e \lambda_i$. If the following inequality holds,

$$(\tau_f - R_{k-1})\mu - \tau_e \lambda_i \geq \mathrm{UB}_{P(\tau_f)} - \textstyle\sum_{r=1}^{k-2} U_r, \tag{5.3}$$

then we still have enough capacity to attain the upper bound even if site i is not visited in tour k. Hence, site i can be filtered out in tour k.

Recall that the sites are indexed in non-decreasing order of the accumulation rate. We can generalize the above inequality to a *Node Filtering Rule* in order to find a threshold value, m^*, such that the first m^* nodes are filtered out in tour k.

Definition 5.1.1. ***(*Node Filtering Rule*)*** *: The vehicle is not allowed to visit sites $i_1, i_2, \ldots, i_{m^*}$ in tour k, where m^* is the largest index such that*

$$(\tau_f - R_{k-1})\mu - \tau_e \textstyle\sum_{l=1}^{m^*} \lambda_{i_l} \geq \mathrm{UB}_{P(\tau_f)} - \textstyle\sum_{r=1}^{k-2} U_r. \tag{5.4}$$

We incorporate the node filtering approach to models M_1 and M_2, and obtain models M_1^F and M_2^F by modifying them as follows. We define $\Lambda_m = \sum_{i=1}^{m} \lambda_i$ and a new decision variable, Z_{mk}.

Z_{mk} : binary variable indicating if every node i with $i \leq m$ is filtered out in tour k $(1 \leq m \leq n, k = 1, 2, \ldots, \kappa_1)$.

Next, we add two additional constraints to the formulation:

$$Y_{ik} \leq 1 - Z_{mk}, \quad \forall i \in N,\ 1 \leq m \leq n,\ i \leq m,\ k = 1, 2, \ldots, \kappa_1, \tag{5.5a}$$

$$M Z_{mk} \geq (\tau_f - R_{k-1})\mu - \mathrm{UB}_{P(\tau_f)} + \textstyle\sum_{r=1}^{k-2} U_r - \tau_e \Lambda_m,\ 1 \leq m \leq n,\ k = 1, 2, \ldots, \kappa_1. \tag{5.5b}$$

Constraint (5.5a) guarantees that if $Z_{mk} = 1$, then a site i, where $i \leq m$, is filtered out in tour $k = 1, 2, \ldots, \kappa_1$. Constraint (5.5b) determines the value of m^* for tour $k = 1, 2, \ldots, \kappa_1$ as defined in the *Node Filtering Rule*.

For the illustrative example provided in Section 4.3, the node filtering approach generates an optimal solution of 0(74.7)-2-1-0(142.1)-2-1-0(221.2)-1-2-4-3-0, with $P(\tau_f) = 1,737$ and $C(\tau_f) = 182$. According to the Constraints (5.5a) and (5.5b), node 4 is filtered out in the tours before τ_e. Although node 3 is not filtered out, the models decide not to visit node 3 before τ_e.

Although the node filtering approach generates practically good results as demonstrated in Section 5.2, it might yield suboptimal solutions as in the case of the two-node CfPP instance illustrated in Figure 5.1. For this instance, suppose that $\tau_e = 60$, $\tau_f = 120$ minutes, and $\mu = 2.6$, $\lambda_1 = 1$, $\lambda_2 = 1.5$ items per minute. The solution found by the node filtering approach is 0(58)-2-0-1-2 with $P(\tau_f) = 111.9$ and $C(\tau_f) = 53$, where node 1 is filtered out in the first tour. On the other hand, an optimal solution for this instance is 0(9)-1-2-0-1-2 with $P^*(\tau_f) = 145.9$ and $C^*(\tau_f) = 102$.

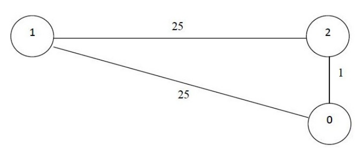


Figure 5.1: A two-node CfPP instance

5.2 Computational experiments

In this section, we analyze the characteristics and performance of the solutions obtained by the proposed solution approach using numerical experiments with realistic data instances. This section starts with a description of the CfPP instances, which are defined based on the data of a clinical laboratory. General results on the performance of our solution approach and detailed analysis on solution characteristics follow. Finally, managerial insights are provided.

5.2.1 Description of the test instances

Using a real-life data set that we have obtained from a clinical testing laboratory that we have been collaborating with, we constructed 12 problem instance sets. Each instance set is defined on a different complete graph, $G = (N^+, E)$, with nodes that correspond to the clients of the clinical laboratory served by a single vehicle in a specified region. The number of nodes in N^+ varies between 10 and 18. The accumulation rate λ_i for each node $i \in N$ is determined according to empirical data such that the majority of the volume comes from a small portion of the clients. The distance d_{ij} for each arc $(i, j) \in E$ is the shortest path distance on the roads obtained from Google Maps. The traveling time, t_{ij}, for each arc $(i, j) \in E$ is taken as $t_{ij} = d_{ij}/\upsilon + s_j$, where υ denotes the constant speed of the vehicle (miles per min) and s_j denotes the service time at node j (in minutes). The service time, s_j, represents the amount of time that the driver spends at node j to pick up the accumulated items. We randomly generated the s_j values from a uniform distribution defined on the interval $[2, 5]$ if $\lambda_j \leq 20$ (items per hour), and on the interval $[6, 9]$ otherwise, based on empirical data.

For each problem instance set, we generated three workload levels ($\alpha_2 = 0.8, 1.0, 1.2$) by varying μ. The value of μ for the corresponding instance is defined using the equality $\tau_e \sum_i \lambda_i = \alpha_2 \mu \tau_f$. As a result, there are 36 data instances in total.

Typically, the sites open at 8 a.m. and close at 7 p.m. Processing at the facility generally ends at 5 a.m. Thus, τ_e is set to 660 minutes, and τ_f is to $1,260$ minutes, considering minutes as the time unit. The initial queue level at the processing facility is set to zero for each instance.

The data sets can be accessed at http://home.ku.edu.tr/eyucel/Research/Dataset CfPP.zip.

The computational experiments were performed on a Xeon E5520 @ 2.27 GHz processing unit with 48GB of memory. GAMS 23.3 was used to create the models. CPLEX 12.2 was employed to solve them with the following options turned on: `threads` 0; `parallelmode` 1. This setting enables CPLEX to run as a multi-threaded application and to distribute the computing load to as many as eight logical cores of the Core i7 Quad-core processor. The run time limit for each M_1 model was set to two hours (7200 s), while each M_2 model was run for four hours (14400 s). A run time limit of four hours (14400 s) was set for each relaxation proposed in Section 4.5.2. In all models, κ_1 was set to 10 as it was observed to be large enough.

5.2.2 Experimental results

In this section, we first compare the solutions found by the proposed models in terms of the processed amount by time τ_f and the transportation cost for a set of instances with different workload levels. Next, we report the solutions of the best performing model with respect to the deviations from the upper bound presented in Section 4.4 on the processed amount and the best lower bound obtained from the two relaxations given in Section 4.5.2 on the transportation cost. In addition, we investigate the effect of the workload level and different accumulation rate patterns on the solutions and the problem difficulty. Finally, we analyze the properties of the best found solutions to derive some managerial insights.

Comparison of the proposed models

By determining the appropriate sites to be filtered out in each tour, the node filtering approach and the assumption of a final TSP tour narrow the search space, and thus, improve the best found solution by the branch-and-bound process within the given time limit. In order to demonstrate its effectiveness, we pick an instance set with 15 nodes and solve the models that seek four different types of solutions: (i) with a final TSP tour and the node filtering constraints (referred to as M_1^{TF} and M_2^{TF}), (ii) without a final TSP tour and with node filtering constraints (M_1^F and M_2^F), (iii) with a final TSP tour and without node filtering constraints (M_1^T and M_2^T), and (iv) without a final TSP tour and node filtering constraints (M_1 and M_2). The best found solution values at the end of a given computation time limit (two hours for model M_1 and six hours for model M_2) are provided in Table 5.1 for varying workload levels of the instance set.

Table 5.1: Best found solution values of the models for the selected instance set

α_2	M_1^{TF}	M_2^{TF}	M_1^T	M_2^T	M_1^F	M_2^F	M_1	M_2
	$P(\tau_f)$	$C(\tau_f)$	$P(\tau_f)$	$C(\tau_f)$	$P(\tau_f)$	$C(\tau_f)$	$P(\tau_f)$	$C(\tau_f)$
0.8	10,493	171	10,493	286	10,493	242	10,493	291
0.9	10,493	212	10,493	260	10,493	270	10,493	292
1.0	10,185	321	10,185	524	10,177	382	10,147	426
1.1	9,269	304	9,269	689	9,269	400	9,247	359
1.2	8,507	284	8,507	582	8,507	330	8,450	296
1.3	7,862	265	7,862	595	7,858	328	7,767	283
1.4	7,307	254	7,307	360	7,307	322	7,281	259

Table 5.1 shows that all models, except M_1, are equally effective in finding the maximal processed amount, $P(\tau_f)$. However, in terms of the transportation cost, $C(\tau_f)$, M_2^{TF} performs the best by far. Based on these observations demonstrating the effectiveness of the node filtering approach with a final TSP tour, the remaining experiments are performed using models M_1^{TF} and M_2^{TF}. We refer to the heuristic approach based on solving these two models within the given time limit as the *TF heuristic* hereafter.

Test results and performance of the TF heuristic

We now evaluate the performance of solutions obtained by the TF heuristic using the upper bound on the processed amount generated by the procedure in Section 4.4 and the lower bound on the transportation cost generated from the MIP relaxations in Section 4.5.2. For each instance, we report the following results.

$P(\tau_f)$	processed amount of the best found solution by the TF heuristic.
$C(\tau_f)$	transportation cost of the best found solution by the TF heuristic.
K	number of tours performed up to time τ_e for the best found solution.
$\overline{v}$	percentage of sites that are visited, averaged over K tours for the best found solution (i.e., $100\sum_{k=1}^{K} v_k/n$, where v_k is the number of sites visited in tour k.)
$\overline{f}$	percentage of sites that are filtered, averaged over K tours for the best found solution (i.e., $100\sum_{k=1}^{K} m_k^*/n$, where the first m_k^* sites are filtered out in tour k.)
DA	achievement percentage, calculated as $100P(\tau_f)/(\tau_e \sum_{i\in N} \lambda_i)$.
DU	capacity utilization percentage, calculated as $100P(\tau_f)/(\mu\tau_f)$.
$\text{Gap}_{P(\tau_f)}$	percentage gap between $P(\tau_f)$ and $\text{UB}_{P(\tau_f)}$, calculated as $100(\text{UB}_{P(\tau_f)} - P(\tau_f))/P(\tau_f)$.

Gap$_{C(\tau_f)}$ percentage gap between $C(\tau_f)$ and $\mathrm{LB}_{C(\tau_f)}$, where $\mathrm{LB}_{C(\tau_f)}$ is the best bound found by CPLEX for the two relaxations at the end of the given run time limit (i.e., $100(C(\tau_f) - \mathrm{LB}_{C(\tau_f)})/\mathrm{LB}_{C(\tau_f)}$.

The experimental results are reported in Table 5.3.

Table 5.3: Test results

Instance set	n	α_2	$P(\tau_f)$	$C(\tau_f)$	K	$\overline{v}$ (%)	$\overline{f}$ (%)	DA (%)	DU (%)	$\text{Gap}_{P(\tau_f)}$ (%)	$\text{Gap}_{C(\tau_f)}$ (%)
1	14	0.8	10,493	171	4	7.14	80.95	100.00	80.00	0.00	0.00
		1.0	10,185	321	9	10.7	59.82	97.06	97.06	1.23	10.69
		1.2	8,507	284	8	11.2	63.27	81.07	97.29	1.07	9.23
2	9	0.8	4,786	143	4	14.8	74.07	100.00	80.00	0.00	0.70
		1.0	4,658	328	9	23.6	68.06	97.33	97.33	1.21	21.93
		1.2	3,886	260	9	16.7	43.06	81.21	97.45	1.16	22.64
3	12	0.8	4,522	173	4	11.1	33.33	100.00	80.00	0.00	3.59
		1.0	4,389	352	10	13	48.15	97.05	97.05	1.27	21.80
		1.2	3,665	306	8	13.1	34.52	81.04	97.25	1.13	15.04
4	13	0.8	4,509	218	3	23.1	42.31	100.00	80.00	0.00	1.40
		1.0	4,361	415	10	13.7	51.28	96.71	96.71	1.56	23.88
		1.2	3,651	362	8	15.4	53.85	80.97	97.16	1.16	20.67
5	14	0.8	4,540	215	3	17.9	42.86	100.00	80.00	0.00	2.38
		1.0	4,362	379	8	16.3	37.76	96.07	96.07	2.30	20.70
		1.2	3,668	351	8	13.3	56.12	80.80	96.96	1.38	21.88
6	11	0.8	4,928	169	3	18.2	72.73	100.00	80.00	0.00	0.00
		1.0	4,746	334	7	21.2	50.00	96.30	96.30	1.93	24.16
		1.2	3,983	320	8	14.3	71.43	80.82	96.98	1.23	24.03
7	11	0.8	4,807	170	3	22.7	59.09	100.00	80.00	0.00	1.19
		1.0	4,643	344	9	18.2	55.68	96.58	96.58	1.46	22.86
		1.2	3,873	290	7	19.7	43.94	80.56	96.67	1.39	20.33
8	12	0.8	4,455	173	3	16.7	75.00	100.00	80.00	0.00	0.58
		1.0	4,301	351	9	16.7	52.08	96.55	96.55	1.74	23.59
		1.2	3,601	312	7	16.7	75.00	80.83	97.00	1.36	22.35
9	11	0.8	4,763	150	3	18.2	54.55	100.00	80.00	0.00	1.35
		1.0	4,613	317	8	18.2	41.56	96.86	96.86	1.48	17.84
		1.2	3,849	270	7	13.6	28.79	80.82	96.98	1.42	6.30
10	16	0.8	5,798	216	4	10.4	52.08	100.00	80.00	0.00	1.89
		1.0	5,592	428	10	11.8	66.67	96.44	96.44	1.92	25.15
		1.2	4,666	378	6	25	50.00	80.47	96.56	1.85	23.93
11	17	0.8	6,214	229	4	13.7	50.98	100.00	80.00	0.00	2.23
		1.0	5,906	387	8	13.4	41.18	95.05	95.05	3.59	24.04
		1.2	4,967	357	9	9.56	42.65	79.94	95.92	2.71	23.96
12	14	0.8	4,687	173	4	11.9	64.29	100.00	80.00	0.00	0.00
		1.0	4,502	311	10	10.3	45.24	96.05	96.05	2.38	21.96
		1.2	3,784	291	8	13.3	40.82	80.74	96.89	1.57	21.76
Avg.					6.78	14.9	53.9	91.51	91.58	1.07	12.97
Min.					3	7.14	28.79	69.63	80.00	0.00	0.00
Max.					10	25	80.95	100.00	97.48	3.59	25.15

We see from Table 5.3 that the gap between the processed amounts of the best found solution by the TF heuristic and the infinite vehicle relaxation solution is 1.07% on the average, while the maximum gap is 3.59%. This shows that this solution approach is very effective in terms of maximizing the processed amount. It also indicates that the infinite vehicle relaxation provides tight upper bounds on the test instances. The optimality gap for the transportation cost, $\text{Gap}_{C(\tau_f)}$, is 12.97% on the average, while the maximum gap is 25.15%. These relatively large gaps may be attributed to several factors: (1) the weakness of the lower bounds, (2) the possibility that the best found solutions may be far from the optimal, (3) inherent computational difficulty of the problem. Similar to researchers studying challenging routing problems such as IRP and TOP [Moin and Salhi, 2007, Archetti et al., 2007], we have observed that finding a good lower bound for CfPP is very difficult. Although our problem relaxations given in Section 4.5.2 provide substantial improvement in achieving stronger bounds, there is still room for further improvement.

Effect of workload level

In order to investigate the effect of the workload level on the solutions and the problem difficulty, we aggregate in Table 5.4 the results by workload level and provide averaged values corresponding to the same workload level. We see that the vehicle performs less number of tours when $\alpha_2 = 0.8$, since a significant portion of the total accumulation can be processed after the final TSP tour. When $\alpha_2 \geq 1.0$, the vehicle performs more tours to prevent idleness of the processing unit during the time interval $(0, \tau_e]$. Correspondingly, the transportation cost becomes higher. When α_2 increases to 1.2, the vehicle starts to perform less number of tours compared to the $\alpha_2 = 1$ case, since it collects a smaller volume up to time τ_e as processing the total accumulation is not possible. In the best found solutions, independent of the workload level, the vehicle visits about 15% of the sites in each tour on the average, implying that it is generally sufficient to visit only a small subset of the nodes to collect "enough" items. In addition, at least 50% of the sites are filtered out on the average in each tour without much loss from optimality. For small α_2, more nodes can be filtered out. For all instances with $\alpha_2 = 0.8$, we see that utilizing 80% of the total processing capacity is sufficient to process all of the total accumulated amount. For the instances with $\alpha_2 \geq 1$, although all the accumulated amount cannot be processed, 96.72% of the total processing capacity is utilized on the average. As expected, the average DA values in Table 5.4 show that the percentage of items processed on time decreases as the workload increases. On the other hand, the processing capacity utilization increases in the workload level. The gaps reported for the best found solutions in terms of both the processed amount and the transportation cost are the highest for $\alpha_2 = 1.0$, which generally represents a difficult case.

Table 5.4: Average results over each workload level

α_2	Avg. $P(\tau_f)$	Avg. $C(\tau_f)$	Avg. K	Avg. $\overline{v}$ (%)	Avg. $\overline{f}$ (%)	Avg. DA (%)	Avg. DU (%)	Avg. $\text{Gap}_{P(\tau_f)}$ (%)	Avg. $\text{Gap}_{C(\tau_f)}$ (%)
0.8	5,375	183	3.50	15.48	58.52	100	80	0	1.28
1.0	5,188	356	8.92	15.59	51.46	96.51	96.51	1.84	21.55
1.2	4,342	315	7.75	15.15	50.29	80.77	96.93	1.45	19.34

Effect of accumulation rate pattern

The accumulation rate pattern of the sites is an influential problem characteristic in CfPP since nodes are selectively visited during the tours before τ_e. To evaluate the performance of the TF heuristic in settings with different accumulation rate patterns, we extend our tests to new data sets. We pick the first instance set in Table 5.3, and in addition to the original pattern (denoted by O), we generate new instance sets by distributing the total accumulation randomly in four patterns: (i) Pareto Random (PR), where 20 percent of sites contribute to 80 percent of the whole accumulation; (ii) Pareto Closer (PC), where 20 percent of sites that are closest to the processing facility accumulate 80 percent of the whole accumulation; (iii) Pareto Distant (PD), where the closest 80 percent of sites contribute to only 20 percent of the whole accumulation; and (iv) Identical (I), where all accumulation rates are identical. In all of the patterns, the sum of the accumulation rates are equal to that of the original one. For the selected problem instance, the geographical locations of the sites (nodes 1 to 14) and the processing facility (node 0) are illustrated in Figure 5.2. The accumulation rates, the traveling times to processing facility, and the service times for each site for pattern O are provided in Table 5.5. In Figure 5.2, the sites are colored such that darker ones have a higher accumulation rate.

The experimental results reported in Table 5.6 indicate that the performance of the TF heuristic, measured by average $\text{Gap}_{P(\tau_f)}$ and $\text{Gap}_{C(\tau_f)}$, decreases in the order of PC, PR, PD, and I. Since the accumulation rates are identical for all sites in pattern I, the filtering rule can filter out only a small percentage of sites (around 10%); thus, the accelerating effect of the node filtering approach on the branch-and-bound algorithm deteriorates. In PC, which resembles the original pattern observed at the clinical laboratory we have been collaborating with, our approach shows the best performance by filtering nearly 50% of the sites. Although the percentage of filtered sites are also nearly 50% in PD, the solution quality gets worse especially in traveling cost ($\text{Gap}_{C(\tau_f)} = 25\%$). While the filtered node percentage is only 34% in pattern PR, our solution approach still provides comparable average gaps (1.3% for the processed amount and 22.2% for the transportation cost).

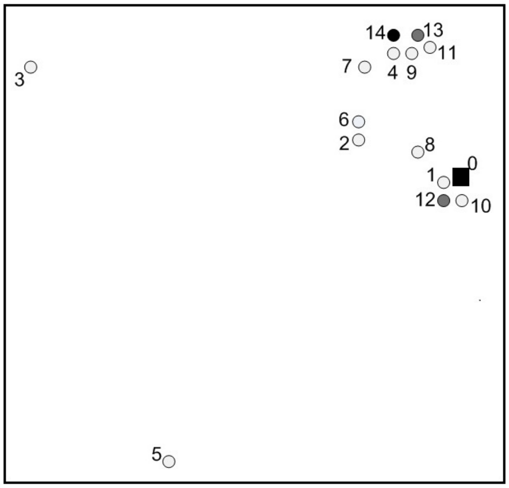


Figure 5.2: The geographical locations of the sites and the processing facility for the selected problem instance set

Table 5.5: Parameters of the selected problem instance set

site i	λ_i (items per min.)	t_{i0} (min.)	λ_i/t_{i0}
1	0.02	1	0.005
2	0.02	7	0.001
3	0.02	8	0.002
4	0.02	10	0.001
5	0.02	11	0.001
6	0.05	8	0.004
7	0.05	11	0.004
8	0.12	3	0.017
9	0.33	10	0.028
10	0.72	2	0.080
11	0.72	10	0.040
12	1.43	2	0.143
13	2.15	10	0.113
14	10.25	9	0.603

In conclusion, we observe that the filtering approach is effective in the Pareto cases, which are probably the most realistic. The extreme case with identical accumulation rates, which is unlikely to be seen in practice, results in relatively poor performance.

Managerial insights

All of the test results up to this point show that both the solution quality and the characteristics of the solution change as the workload level changes. In this section, we further investigate the real-life implications of this factor by additional tests with the aim of providing some generalizable insights. We pick, again, the first instance set in Table 5.3 and solve it for a number of different α_2 values varying between 0.8 and 1.4 (obtained by decreasing the processing rate μ while keeping the total accumulated amount constant).

Table 5.6: Test results for the selected problem instance with different accumulation rate patterns

Pattern	α_2	$P(\tau_f)$	$C(\tau_f)$	K	$\overline{v}$ (%)	$\overline{f}$ (%)	DA (%)	DU (%)	$\text{Gap}_{P(\tau_f)}$ (%)	$\text{Gap}_{C(\tau_f)}$ (%)
PC	0.8	10,493	154	3	9.52	59.67	100.00	80.00	0.00	0.00
	1.0	10,315	308	10	25	53.00	98.30	98.30	0.85	21.26
	1.2	8,612	249	9	22.2	39.51	82.07	98.48	0.66	5.51
	Avg.	9,807	237	7.33	18.92	50.72	93.46	92.26	0.50	8.92
PR	0.8	10,493	169	3	16.7	28.57	100.00	80.00	0.00	0.00
	1.0	10,192	381	8	31.3	50.00	97.13	97.13	1.99	39.05
	1.2	8,511	305	9	38.1	25.40	81.11	97.33	1.90	27.62
	Avg.	9,732	285	6.67	28.67	34.66	92.75	91.49	1.30	22.22
PD	0.8	10,493	223	4	16.1	65.75	100.00	80.00	0.00	0.00
	1.0	10,035	419	8	31.3	59.25	95.63	95.63	2.56	43.00
	1.2	8,391	367	6	39.3	25.07	79.97	95.96	2.31	32.01
	Avg.	9,640	336	6.00	28.87	50.02	91.87	90.53	1.62	25.01
I	0.8	10,493	303	3	85.7	7.14	100.00	80.00	0.00	27.31
	1.0	9,891	511	9	23	11.11	94.26	94.26	4.69	98.06
	1.2	8,323	451	10	26.4	11.43	79.32	95.18	3.78	81.85
	Avg.	9,569	422	7.33	45.05	9.89	91.19	89.81	2.82	69.08

According to the results presented in Table 5.7, we observe that all the accumulated amount can be processed when $\alpha_2 < 1.0$. As the workload level increases, the percentage of the total accumulated amount that can be processed on time decreases,

Table 5.7: Test results for the selected problem instance for varying workload levels

α_2	$P(\tau_f)$	$C(\tau_f)$	K	$\overline{v}$ (%)	$\overline{f}$ (%)	DA (%)	DU (%)	$\text{Gap}_{P(\tau_f)}$ (%)	$\text{Gap}_{C(\tau_f)}$ (%)
0.8	10,493	171	4	7.14	80.95	100.00	80.00	0.00	0.00
0.9	10,493	212	5	10.7	44.64	100.00	90.00	0.00	1.44
1.0	10,185	321	9	10.7	59.82	97.06	97.06	1.23	10.69
1.1	9,269	304	8	11.2	63.27	88.34	97.17	1.16	9.35
1.2	8,507	284	8	11.2	63.27	81.07	97.29	1.07	9.23
1.3	7,862	265	9	9.52	55.95	74.92	97.40	1.01	1.53
1.4	7,307	254	7	8.33	69.05	69.63	97.48	0.96	0.40

whereas the processing capacity utilization increases. The gap between the processed amount of the solution and the corresponding infinite vehicle relaxation solution is the largest when $\alpha_2 = 1.0$ and it decreases as the workload exceeds 1.

When $\alpha_2 < 1.0$, idle times at the processing unit are tolerable during $(0, \tau_e]$ since most of the items can be processed after the final TSP tour. Since a smaller number of tours is sufficient, the best found transportation costs are low. When $\alpha_2 = 0.8$, the problem becomes relatively easy as around 81% of nodes can be filtered without causing any significant loss from optimality. As α_2 increases to 0.9, it is more difficult to identify the nodes to be filtered ($\overline{f} = 44.64\%$). In addition, the percentage of nodes visited in a tour increases from 7.14% to 10.7%, causing the transportation cost to increase by 24%. In case of $\alpha_2 = 1.0$, the problem becomes significantly more difficult since the number of tours required almost doubles. The vehicle visits the same percentage of nodes in a tour but has to perform more frequent tours to ensure that the processing center is well-supplied with the items to be processed. The processing rate decreases as α_2 increases, so that it requires more time to process the same amount of items. As a result, the best found transportation cost decreases in α_2.

When $\alpha_2 \geq 1.0$, capacity utilization remains constant due to routing restrictions. For $\alpha_2 = 1.4$, both M_1 and M_2 can be solved to near-optimality since filtering becomes more effective while both K and $\overline{v}$ decrease.

The best found solutions by the TF heuristic for each workload level are provided in Table 5.8, where the numbers in parentheses denote the amount of waiting times at the corresponding nodes. The TSP tour performed after time τ_e is 8-6-2-7-14-5-11-13-4-9-3-12-10-1 with a cost of 108. According to these solutions, before the first tour a longer waiting time can be tolerated when $\alpha_2 < 1$, since all the accumulated amount can be processed easily on time. For larger workload levels, the waiting time before the first tour is smaller. For all workload levels, it is better to visit closer sites with high accumulation rates (i.e., nodes 12, 13, and 14) before τ_e. In none of the solutions, nodes 1 to 7, which have the lowest accumulation rates, are visited before τ_e. For these nodes, the ratio of the accumulation rate to the total time to service a node and return to the processing facility (given in Table 5.5) is less than 0.01.

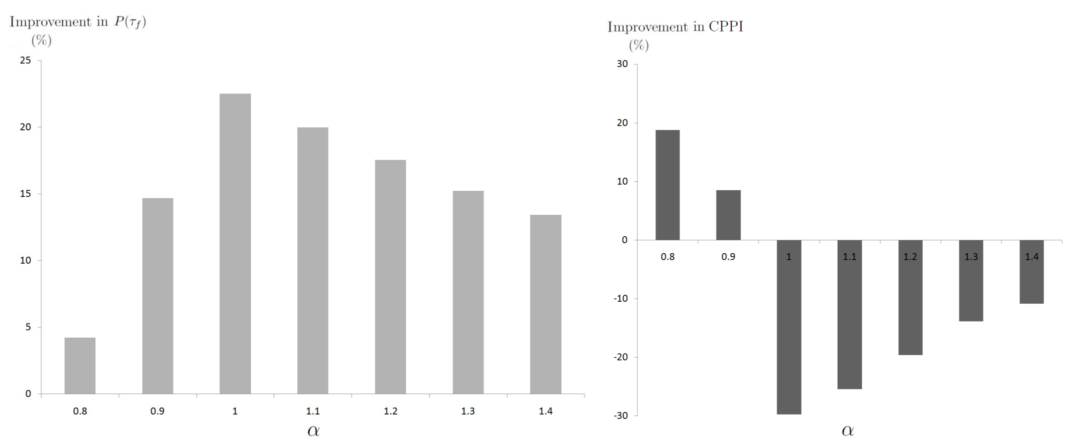
In Table 5.8, we report the idle time of the processing facility throughout the day and the idle time of the vehicle during the working day (i.e., between times 0 and τ_e). As the workload level increases, the idle times of the processor decrease, while the vehicle stays almost at the same level of idleness for $\alpha_2 \geq 1$.

Next, we compare these solutions with the solutions obtained by a simple solution approach. In real life, practitioners tend to apply simpler heuristic approaches to determine the route and schedule of the vehicles. As an example, for instances like the selected instance set, in which a small percentage of sites collects a high percentage of the total accumulation, a common simple solution approach used by the practitioners is to perform two tours before τ_e, one in the morning and one in the afternoon,

Table 5.8: The best found solutions for the selected instance for varying workload levels

α_2	Best Found Solution	Processor Idle Time (min.)	Vehicle Idle Time (min.)
0.8	0(226)-14-0(161.42)-10-0(61.58)-14-0(57)-TSP-0	252	506
0.9	0(100)-14-0(85.35)-12-0(16.43)-14-13-9-0(19.28)-14-0(64.94)-TSP-0	126	465
1.0	0-13-14-0(1.2)-14-0(7.47)-14-0(11.2)-10-12-0-9-13-14-0(101.18)-14-0(14.53)-12-0(29.04)-14-0-TSP-0	37	294.6
1.1	0(0.66)-8-9-14-0-14-0(12.28)-10-12-0-14-0(33.9)-14-0(63.1)-13-14-0(181.35)-14-0-TSP-0	35.7	291.3
1.2	0(4.15)-9-14-0-14-0(1.4)-13-14-0(58.71)-10-12-0(25.16)-14-0(180.81)-10-0(0.03)-14-0-TSP-0	34.2	270.3
1.3	0(2.84)-9-14-0-14-0(4.6)-13-14-0(68.17)-12-0(23.2)-14-0(181)-14-0-TSP-0	32.8	279.8
1.4	0(1.71)-9-14-0-14-0(17.91)-14-0(64.55)-12-0(14.51)-14-0(175.72)-14-0-TSP-0	31.7	274.4

that visit only a certain percentage of closer sites with high accumulation rate. The solution proposed by the simple policy for the selected instance set is to make one tour at time 120 and one tour at time 420, both of which visit only sites 10, 12, and 14, which are among 20% of sites with the highest λ_i/t_{i0} ratio. After τ_e, the vehicle performs a TSP tour to collect the remaining items from sites. This solution has a cost of 202 for all workload levels. Figure 5.3a provides the percentage of improvement on the processed amount and Figure 5.3b provides the percentage of improvement on cost per processed item (CPPI) provided by our solution approach compared to the simple policy for different workload levels. It is observed that the simple policy cannot process all the accumulated amount even for the smallest workload level, whereas our solution approach provides a solution that processes all of the collected items with a smaller transportation cost of 171. As α_2 increases, the costs of proposed solutions are larger than that of the simple policy but the improvement percentages on the processed amount are significant. In case of $\alpha_2 = 1$, the percentage improvement provided by our solution approach on the processed amount is 22.5%. However, as seen in Figure 5.3b, the improvement in customer service comes at an additional cost.



(a) on the processed amount **(b)** on the cost per processed item

Figure 5.3: The percentage improvement provided by our solution approach compared to the simple policy for different workload levels for the selected instance set

As a result, we can provide the following insights to the decision makers. First, for all workload levels and accumulation rate patterns, it is not efficient to perform tours that visit all sites before τ_e. Second, if the accumulation rates of sites show a

Pareto distribution, regardless of the place of processing facility, a significant portion of the sites can be filtered out in each tour before τ_e. Third, as the workload level increases, the vehicle should perform more tours visiting only a small subset of sites, which are closer and have higher accumulated amounts. Fourth, the performance of simple solution approaches depends on the workload level, and these approaches might yield significantly suboptimal solutions.

5.3 Conclusions and summary of contributions

In this chapter, we propose an MIP-based heuristic approach for the single vehicle collection for processing problem with two prioritized objectives. The proposed heuristic is a lexicographic approach, since it first optimizes the objective with the first priority, and then, it optimizes the objective with the second priority among all alternative optimal solutions. The heuristic incorporates additional constraints to the MIP model with the goal of eliminating solutions that are likely to be suboptimal. The computational tests show that the proposed approach finds solutions that are very close to optimal in terms of the processed amount. Furthermore, we show that transportation costs of these solutions are in the vicinity of 13% to optimality on the average.

We identify the workload level (i.e., α_2) as a key indicator parameter for problem difficulty. For small ($\alpha_2 < 1$) and large ($\alpha_2 \geq 1.2$) workload levels, the problem becomes easier. When α_2 is close to 1, the vehicle should perform frequent, and more carefully designed tours in order to feed the processor continuously. In such cases, a high service level can be achieved at the expense of a higher transportation cost. Another indicator of problem difficulty is the pattern of the accumulation at the sites. When the pattern is Pareto visiting sites with low accumulation rates can be ruled out without significantly compromising from optimality.

Chapter 6

A PRIORITIZED BICRITERIA HEURISTIC FOR THE SINGLE VEHICLE PROBLEM

In this chapter, we focus on the single vehicle collection for processing problem with two prioritized objectives and present a heuristic algorithm that obtains high quality solutions even for larger-size problem instances in reasonable computation time. Since our heuristic approach combines a tabu search algorithm by a prioritized search mechanism and solves an LP model to evaluate each candidate solution at every iteration, it is referred to as BM (bicriteria matheuristic).

The differences between the proposed tabu search based matheuristic, we propose in this chapter, and a classical tabu search approach are as follows: Our BM algorithm addresses a bicriteria problem with priority levels. In order for this, the algorithm evaluates a candidate solution in terms of both objectives and determines the next iteration's solution according to the priorities associated with each objective. Our BM algorithm can be considered as a matheuristic approach [see Maniezzo et al., 2009] due to its hybridization approach that is based on the integration of LP with a tabu search scheme. The solution with either first improvement or best improvement is selected according to the neighborhood solutions.

The remainder of this chapter is organized as follows. The proposed solution methodology is presented in Section 6.1. Results and analysis of numerous computational experiments conducted on realistic data instances are provided in Section 6.2. Finally, Section 6.3 concludes with a summary and contributions of this chapter.

6.1 Solution approach

More specifically, our BM algorithm uses a tabu search algorithm to determine the sequence of nodes to be visited in all tours, i.e., the giant-tour, while the node visit times of a giant-tour are scheduled through a linear programming (LP) model. The LP model optimally schedules each giant-tour examined by the tabu search algorithm under the single objective of maximizing the processed amount by time τ_f. The tabu search searches the solution space in order to find a solution that processes as many items as possible until the deadline, with the first priority, and to minimize the transportation cost for the best found processed amount, with the second priority. Upon termination, the algorithm provides a feasible scheduled giant-tour.

Two properties of optimal solutions to CfPP have been characterized by Propositions 4.3.2 and 4.3.3 in Section 4.3.2. The former states that any solution in which the vehicle waits at a site can be converted to one in which the vehicle waits only at the processing facility, without suboptimality. The latter states that there exists an optimal solution such that a site is visited at most once in each tour. The BM seeks solutions with these two properties.

The description of the BM algorithm is organized into the following sections. Section 6.1.1 provides the solution representation and the notation. Section 6.1.2 describes a simple constructive heuristic that is used in the initial solution construction. The LP model that finds an optimal schedule for a partial solution (a giant-tour) is given in Section 6.1.3. Section 6.1.4 presents the tabu search algorithm that guides its search according to the two prioritized objectives. Finally, the general scheme of the BM algorithm is given in Section 6.1.5.

6.1.1 Solution representation and notation

As described in Section 3.2, a CfPP solution has two components: a giant-tour and a schedule of the giant-tour.

The giant-tour component of a solution $g \in \mathcal{G}$, which is a partial solution, is represented by a matrix, π, with $\kappa_1 + \kappa_2$ rows and $n + 1$ columns. The first κ_1 rows of π correspond to D-tours and the remaining κ_2 tours correspond to A-tours. An element in a row is a leg in a tour such that the vehicle visits the j^{th} element in the j^{th} order. Since the vehicle is not required to visit all n nodes in a tour, there might be a consecutive series of empty elements at the end of the corresponding row. In addition, empty tours are also allowed.

In the algorithm, the schedule of a giant-tour is uniquely determined by an optimal solution found by the LP model. Therefore, in the BM algorithm there is a one-to-one correspondence between g and π.

6.1.2 Construction of an initial solution

An initial solution is required to start the search procedure. For this purpose, first, a simple constructive tour building heuristic, which we refer to as TBH in short, is applied to obtain an initial feasible scheduled giant-tour. Next, the schedule of the initial solution is optimized in terms of the processed amount through the LP model provided in Section 6.1.3. In the following paragraphs, we provide a detailed description of the TBH algorithm.

The TBH constructs tours one at a time, dynamically, with the objective of minimizing the idleness of the processor during each tour, hence, in turn maximizing the processed amount by time τ_f.

Let tour k be the current tour to be constructed. The TBH, first, determines an upper bound for the duration of the tour, denoted by Δ_k, so that the processor does not stay idle during the time interval $(R_{k-1}, R_k]$. Note that, the queue level at the processing facility at the return time from tour $k-1$, is $Q(R_{k-1})$, and the time required for depletion of the queue at the processing facility is $Q(R_{k-1})/\mu$. If tour k is completed by time $R_{k-1} + Q(R_{k-1})/\mu$, there will be no idle time at the processing facility during the tour. Accordingly, the TBH sets $\Delta_k = Q(R_{k-1})/\mu$. However, the minimum possible tour length might be greater than Δ_k, i.e., $\min_{i \in N}\{t_{0i} + t_{i0}\} > \Delta_k$, then the processor has to stay idle for some time. In this case, the tour is constructed according to a greedy rule, referred to as the *Collection Rate Rule*, which aims to maximize the amount of items collected per unit time during the current tour. On the other hand, if it is possible to construct a tour that returns to the processing facility within Δ_k time units, the TBH constructs the tour according to the *Non-Idling Rule*, which aims to collect as many items as possible within Δ_k time units.

Before explaining these rules in details, let us provide some notation for the current tour k. Let $V_k \subseteq N$ denote the set of sites visited, and node $i \in V_k$ be the last node visited so far in tour k. Let ε_{pk} represent the elapsed time from the processing facility to node $p \in V_k$ and a_{pk} represent the accumulated amount at site $p \in V_k$ at the time of visit. Note that both the time passed since the last visit of the site and whether the visit time is before τ_e or after τ_e affects the accumulated amount. Then, since $I_p(R_{k-1})$ denotes the amount of items at node p at time R_{k-1}, $a_{pk} = I_p(R_{k-1}) + \lambda_p(\min\{R_{k-1} + \varepsilon_{pk}, \tau_e\} - \min\{R_{k-1}, \tau_e\})$.

Starting with an empty tour, i.e., $V_k = \emptyset$, the current node $i = 0$, and $\varepsilon_{ik} = 0$, both the *Collection Rate Rule* and the *Non-Idling Rule* determine the nodes to be appended at the end of tour k, one at a time, as follows.

Each time the *Collection Rate Rule* appends node j^*, which is not visited so far, i.e., $j^* \in N \backslash V$, and maximizes the amount of items collected per unit time. More specifically,

$$j^* = \underset{j \in N^+ \setminus V_k}{\operatorname{argmax}} \left\{ \frac{\sum_{p \in V_k} a_{pk} + a_{jk}}{\varepsilon_{ik} + t_{ij} + t_{j0}} \right\},$$

where a_{jk} is the accumulated amount at site j by the time of visit, which is calculated as $a_{jk} = I_j(R_{k-1}) + \lambda_j(\min\{R_{k-1} + \varepsilon_{ik} + t_{ij}, \tau_e\} - \min\{R_{k-1}, \tau_e\})$. Node j^* is added to V_k setting $\varepsilon_{j^*k} = \varepsilon_{ik} + t_{ij^*}$. (Note that j^* might be the processing facility, where $a_{0k} = 0$.) The construction of tour k continues until $j^* = 0$, in which case returning to the processing facility results in the largest amount of items collected per unit time, or, $V_k = N$, in which case all of the sites are visited.

Each time the *Non-Idling Rule* appends node j^*, which is not visited so far, i.e., $j^* \in N \setminus V$, and satisfies $\varepsilon_{ik} + t_{ij^*} + t_{j^*0} \leq \Delta_k$, and has the largest accumulated amount per unit additional travel time spent to obtain that amount. More specifically,

$$j^* = \underset{j \in N \setminus V_k | \varepsilon_{ik} + t_{ij} + t_{j0} \leq \Delta_k}{\operatorname{argmax}} \left\{ \frac{a_{jk}}{\delta_{ij}} \right\},$$

where a_{jk} is the accumulated amount at site j at the time of visit, like in the case of the *Collection Rate Rule*, and δ_{ij} is the additional traveling time spent for visiting site j, instead of returning to the processing facility right after visiting site i, i.e., $\delta_{ij} = t_{ij} + t_{j0} - t_{i0}$. Node j^* is added to V_k setting $\varepsilon_{j^*k} = \varepsilon_{ik} + t_{ij^*}$. The construction of tour k continues as long as there exists a feasible candidate node j with positive item volume, i.e., $a_{jk} > 0$. Note that the tour duration, i.e., $\varepsilon_{ik} + t_{i0}$, should be less than or equal to Δ_k. If $\varepsilon_{ik} + t_{i0}$ is strictly less than Δ_k, then the algorithm inserts a waiting time of $W_k = \Delta_k - (\varepsilon_{ik} + t_{i0})$ time units at the processing facility before starting the tour. Note that inserting a positive waiting time before the tour increases the collected amount from every visited site.

After constructing tour k, the algorithm constructs the next tour, tour $k + 1$, and so on. The TBH continues until the current time reaches time τ_f or all of the items accumulated at sites are delivered to the processing facility. The pseudocode of the

TBH is provided as follows.

By construction, the TBH prioritizes maximizing the processed amount and only indirectly aims to minimize the transportation cost. Therefore, although the solutions found by the TBH are effective in terms of the processed amount, they incur high transportation cost as observed through computational experiments provided in Section 6.2. Anyway, the BM algorithm uses the TBH as a fast heuristic that provides good initial solutions.

Note that the TBH algorithm has no restriction on the number of tours of the resulting solution. However, the BM algorithm constructs solutions that have at most κ_1 tours before τ_e and at most $\kappa_2 = n$ tours after τ_e. Therefore, if the TBH solution has more than κ_1 tours before τ_e, then κ_1 is increased to the number of D-tours of the initial solution generated by TBH. The impact of setting κ_1 in such a dynamic way is examined in Section 6.2.3.

Algorithm 1 TBH algorithm

Set $k \leftarrow 1$, $R_0 \leftarrow 0$, $I_j(R_0) \leftarrow 0$ for all $j \in N$.
while $R_{k-1} < \tau_f$ **do**
  Set $i \leftarrow 0$, $V_k \leftarrow \{\}$, $W_k \leftarrow 0$, $\varepsilon_{jk} \leftarrow 0$ for all $j \in N$, $\Delta_k \leftarrow Q(R_{k-1})/\mu$.
  if $\min_{i \in N}\{t_{0i} + t_{i0}\} > \Delta_k$ **then** // *Collection Rate Rule*
    Set $j^* \leftarrow \text{argmax}_{j \in N^+ \setminus V_k}\{(\sum_{p \in V_k} a_{pk} + a_{jk})/(\varepsilon_{ik} + t_{ij} + t_{j0})\}$ where $a_{jk} = I_j(R_{k-1}) + \lambda_j(\min\{R_{k-1} + \varepsilon_{ik} + t_{ij}, \tau_e\} - \min\{R_{k-1}, \tau_e\})$.
    while $j^* \neq 0$ and $V \subset N$ **do**
      Set $V_k \leftarrow V_k \cup \{j^*\}$, $\varepsilon_{j^*k} \leftarrow \varepsilon_{ik} + t_{ij^*}$, $i \leftarrow j^*$.
      Set $a_{j^*k} \leftarrow I_{j^*}(R_{k-1}) + \lambda_{j^*}(\min\{R_{k-1} + \varepsilon_{j^*k}, \tau_e\} - \min\{R_{k-1}, \tau_e\})$.
      Set $j^* \leftarrow \text{argmax}_{j \in N^+ \setminus V_k}\{(\sum_{p \in V_k} a_{pk} + a_{jk})/(\varepsilon_{ik} + t_{ij} + t_{j0})\}$.
    end while
  else// *Non-Idling Rule*
    Set $j^* \leftarrow \text{argmax}_{j \in N \setminus V_k | \varepsilon_{ik} + t_{ij} + t_{j0} \leq \Delta_k}\{a_{jk}/\delta_{ij}\}$.
    while $a_{j^*k} > 0$ **do**
      Set $V_k \leftarrow V_k \cup \{j^*\}$, $\varepsilon_{j^*k} \leftarrow \varepsilon_{ik} + t_{ij^*}$, $i \leftarrow j^*$.
      Set $a_{j^*k} \leftarrow I_{j^*}(R_{k-1}) + \lambda_{j^*}(\min\{R_{k-1} + \varepsilon_{j^*k}, \tau_e\} - \min\{R_{k-1}, \tau_e\})$.
      Set $j^* \leftarrow \text{argmax}_{j \in N \setminus V_k | \varepsilon_{ik} + t_{ij} + t_{j0} \leq \Delta_k}\{a_{jk}/\delta_{ij}\}$.
    end while
    Set $W_k = \max\{\Delta_k - (\varepsilon_{ik} + t_{i0}), 0\}$.
  end if
  Set $R_k \leftarrow R_{k-1} + W_k + \varepsilon_{ik} + t_{i0}$.
  Set $Q(R_k) = Q(R_{k-1}) - \min\{Q(R_{k-1}), \mu(R_k - R_{k-1})\} + \sum_{j \in V_k}(I_j(R_{k-1}) + \lambda_j(\min\{R_{k-1} + W_k + \varepsilon_{jk}, \tau_e\} - \min\{R_{k-1}, \tau_e\}))$.
  Set $I_j(R_k) \leftarrow I_j(R_{k-1}) + \lambda_j(\min\{R_k, \tau_e\} - \min\{R_{k-1}, \tau_e\})$ for all $j \in N \setminus V_k$.
  Set $I_j(R_k) \leftarrow \lambda_j(\min\{R_k, \tau_e\} - \min\{R_{k-1} + W_k + \varepsilon_{jk}, \tau_e\})$ for all $j \in V_k$.
  Set $k \leftarrow k + 1$.
end while

6.1.3 The LP model for scheduling

For a given partial CfPP solution (a sequence of nodes, i.e., a giant-tour), the scheduling problem aims to provide a complete CfPP solution (a scheduled giant-tour) by scheduling the giant-tour with the single objective of maximizing the processed amount by time τ_f. As the vehicle is allowed to wait only at the processing facility according to Proposition 4.3.2, scheduling the node visit times of a giant-tour reduces to finding the waiting time at the processing facility prior to each tour. There are several possible approaches to solve this scheduling problem. For example, a metaheuristic algorithm such as a tabu search or a simulated annealing might be implemented for this purpose. However, since all the expressions for the objective function and the constraints are linear, and we have nonnegative decision variables, which are not restricted to be integer, in this problem, the fastest way of obtaining an exact solution is using an LP model. Therefore, we developed the following LP formulation to schedule a giant-tour.

The model maximizes the processed amount by time τ_f for a given giant-tour π and determines the optimal schedule of π, the node visit times in each tour, i.e., T_{ik}'s. The model has the following parameters and decision variables.

Parameters

x_{ijk} binary variable indicating if node j is visited immediately after node i in tour k, $(i, j \in N,\ k = 1, 2, \ldots, \kappa_1 + \kappa_2)$. If $x_{00k} = 1$, then tour k is an empty tour.

ε_{ik} required time to go from node 0 to node i in tour k, $(i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2)$. If node i is not visited in tour k, then $\varepsilon_{ik} = 0$.

Decision variables

T_{ik} visit time of node i at tour k ($i \in N^+, k = 1, 2, \ldots, \kappa_1 + \kappa_2$). If node i is not visited in tour k, it denotes the last visit time of node i before tour k (i.e., $T_{ik} = T_{i,k-1}$). T_{0k} denotes the starting time of tour k.

R_k return time of the vehicle from tour k to processing facility ($k = 1, \ldots, \kappa_1 + \kappa_2$); $R_0 = 0$.

E_{ik} collected amount from site i in tour k, ($i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

S_k total collected amount in tour k, ($k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

L_{ik} auxiliary variable used to calculate E_{ik} ($i \in N$, $k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

W_k waiting time of the vehicle at processing facility at the beginning of tour k, ($k = 1, 2, \ldots, \kappa_1 + \kappa_2$).

Q_k amount of unprocessed items at the processing facility at the end of tour k, ($k = 0, 1, 2, \ldots, \kappa_1 + \kappa_2$). Q_0 is a parameter, which represents the number of unprocessed items at the beginning of the day.

U_k amount of processed items between R_k and R_{k+1}, ($k = 0, 1, 2, \ldots, \kappa_1 + \kappa_2 - 1$).

$U_{\kappa_1+\kappa_2}$ amount of processed items between $R_{\kappa_1+\kappa_2}$ and τ_f.

Model

$$M\text{: Maximize} \quad \sum_{k=1}^{\kappa_1+\kappa_2} U_k$$

subject to

$$R_k = R_{k-1} + W_k + \sum_{(i,j)\in A} t_{ij} X_{ijk}, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{6.1a}$$

$$T_{0k} = R_{k-1} + W_k, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{6.1b}$$

$$R_{\kappa_1+\kappa_2} \leq \tau_f, \tag{6.1c}$$

$$T_{0,\kappa_1} \leq \tau_e, \tag{6.1d}$$

$$T_{0,\kappa_1+1} \geq \tau_e, \tag{6.1e}$$

$$\begin{aligned} T_{ik} = &\sum_{j\in N} x_{kij}(R_{k-1} + W_k + \varepsilon_{ik}) \\ &+ (1 - \sum_{j\in N} x_{kij})T_{i,k-1}, \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2 \end{aligned} \tag{6.1f}$$

$$L_{ik} \leq T_{ik}, \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{6.2a}$$

$$L_{ik} \leq \tau_e, \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{6.2b}$$

$$E_{ik} = \lambda_i(L_{ik} - L_{i,k-1}), \qquad \forall i \in N, k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{6.2c}$$

$$L_{i0} = 0, \qquad \forall i \in N, \tag{6.2d}$$

$$S_k = \sum_{i\in N} E_{ik}, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{6.2e}$$

$$\sum_{k=1}^{\kappa_1+\kappa_2} E_{ik} = \lambda_i \tau_e, \qquad \forall i \in N, \tag{6.2f}$$

$$U_k \leq \mu(R_{k+1} - R_k), \qquad \forall k = 0, 1, \ldots, \kappa_1 + \kappa_2 - 1, \tag{6.3a}$$

$$U_{\kappa_1+\kappa_2} \leq \mu(\tau_f - R_{\kappa_1+\kappa_2}), \tag{6.3b}$$

$$U_k \leq Q_k, \qquad \forall k = 0, 1, \ldots, \kappa_1 + \kappa_2, \tag{6.3c}$$

$$Q_k = Q_{k-1} - U_{k-1} + S_k, \qquad \forall k = 1, 2, \ldots, \kappa_1 + \kappa_2, \tag{6.4}$$

$$\text{All variables nonnegative.} \tag{6.5}$$

Constraints (6.1a)-(6.1e) define the relation between tour return times, waiting times, and tour start times. Constraints (6.1a) and (6.1b) calculate the return time and the starting time of a tour, respectively. Constraint (6.1c) ensures that the last tour should be completed by τ_f, and Constraints (6.1d)-(6.1e) limit the number of tours before τ_e by κ_1.

Constraint (6.1f) defines the node visit times. If the vehicle visits node j immediately after node i in tour k, the constraint restricts the vehicle to be at node j at time $T_{ik} + t_{ij}$. If node i is not visited in tour k, T_{ik} equals the last visit time before that tour, i.e., $T_{i,k-1}$.

Constraints (6.2a)-(6.2f) are used to calculate the collected amounts from the visited nodes. As the accumulation at sites ends at time τ_e, $\lambda_i(\min\{T_{ik}, \tau_e\} - \min\{T_{i,k-1}, \tau_e\})$ gives the collected amount from node i in tour k. Note that, if node i is not visited in tour k, $\lambda_i(\min\{T_{ik}, \tau_e\} - \min\{T_{i,k-1}, \tau_e\})$ is zero. We linearize this quantity by defining the auxiliary variables, L_{ik}'s, as $\min\{T_{ik}, \tau_e\}$ through Constraints (6.2a)-(6.2b). Constraints (6.2c)-(6.2e) calculate the collected amount in tour k. Constraint (6.2f) ensures that all of the items accumulated at the nodes are collected.

The amount of processed items between the return times of two consecutive tours, U_k, cannot be greater than the amount of unprocessed items, Q_k, and the processing capacity, $\mu(R_{k+1} - R_k)$. This condition is satisfied via Constraints (6.3a)-(6.3c).

Constraint (6.4) balances the queue size at the processing facility.

Finally, Constraint (6.5) dictates nonnegativity of all variables.

For a given giant-tour, Model M can be used to generate a feasible scheduling plan with a maximum processed amount by time τ_f.

6.1.4 The tabu search algorithm for routing

The BM algorithm utilizes a bicriteria tabu search technique to search the solution space in order to find a solution that maximizes the processed amount by a deadline, with the first priority, and to minimize the transportation cost for the best found processed amount, with the second priority. The tabu search algorithm makes use of the notation provided in Table 6.3.

The tabu search algorithm starts by setting the giant-tour of the solution of the TBH algorithm as the current giant-tour and keeps moving from the current giant-

Table 6.3: Notation used for the description of the tabu search algorithm of the BM

π	current giant-tour
π^*	best found giant-tour so far
π'	neighboring giant-tour of giant-tour π, $\pi' \in \mathcal{N}(\pi)$
$\mathcal{N}_{\mathrm{I}}(\pi)$	neighboring giant-tours of giant-tour π with a larger processed amount compared to that of π.
$\mathcal{N}_{\mathrm{II}}(\pi)$	neighboring giant-tours of giant-tour π with the same processed amount and smaller transportation cost compared to those of π.
$\mathcal{N}_{\mathrm{III}}(\pi)$	neighboring giant-tours of giant-tour π with the same processed amount and the same or larger transportation cost compared to those of π.
$\mathcal{N}_{\mathrm{IV}}(\pi)$	neighboring giant-tours of giant-tour π with a smaller processed amount compared to that of π.
π_{I}	a giant-tour in $\mathcal{N}_{\mathrm{I}}(\pi)$
π^*_{II}	$\pi^*_{\mathrm{II}} \in \mathcal{N}_{\mathrm{II}}(\pi)$ that has the smallest transportation cost
π^*_{III}	$\pi^*_{\mathrm{III}} \in \mathcal{N}_{\mathrm{III}}(\pi)$ that has the smallest transportation cost
π^*_{IV}	$\pi^*_{\mathrm{IV}} \in \mathcal{N}_{\mathrm{IV}}(\pi)$ that has the largest processed amount
η	total number of iterations allowed
γ	number of iterations allowed without an improvement in any of the objective functions
θ	tabu tenure

tour π to a neighboring giant-tour $\pi' \in \mathcal{N}(\pi)$ until a stopping condition is reached.

Neighborhood definition: $\mathcal{N}(\pi)$ is composed of all the giant-tours that can be reached by performing one of the following four move operations to the current giant-tour, π. While performing these operations, feasibility due to the constraints that (i) every tour visits a node at most once, (ii) each node must be visited exactly once after time τ_e, (iii) the last D-tour should start before time τ_e, and (iv) the first node of the first A-tour should be visited after τ_e should be retained.

1. ***Swap*:** Two nodes visited after τ_e are swapped. Since exactly n nodes are visited after τ_e, π has at most $n(n-1)/2$ neighbors due to *Swap* move operation.

2. ***Delete*:** A node visited in a D-tour is deleted from that tour. In any D-tour, at most n nodes can be visited, meaning that π has at most $n\kappa_1$ neighbors due to *Delete* move operation.

3. ***Insert*:** An unvisited node is inserted as the first leg of a D-tour. Since the set of unvisited sites of a tour can have at most n members, *Insert* move operation results in at most $n\kappa_1$ neighbors.

4. ***Mutate*:** A node i visited in a D-tour is deleted from that tour and a node j, which is selected randomly among the unvisited nodes in that tour, is inserted in place of node i. In any D-tour, at most n nodes can be visited, meaning that π has at most $n\kappa_1$ neighbors due to *Mutate* move operation.

The size of $\mathcal{N}(\pi)$ is $O(n^2 + n\kappa_1)$.

Tabu lists: When a node is removed from a tour through *Swap*, *Delete*, or *Mutate* move operations, inserting it again into the same tour through *Swap*, *Insert*, or *Mutate* move operations is prohibited, declared tabu, for θ iterations. Conversely, when a node is inserted to a tour through *Swap*, *Insert*, or *Mutate* move operations, removing it again from the same tour through *Swap*, *Delete*, or *Mutate* move operations is prohibited, for θ iterations. Similarly, when two nodes visited after τ_e are swapped, exchanging them again is prohibited for θ iterations. Therefore, each move operation has its own tabu list with size θ.

As an aspiration criterion, the BM algorithm allows solutions which are better than the currently-known best solution in terms of the processed amount.

Solution evaluation: Since the LP model is used to schedule a giant-tour optimally, it provides the optimal processed amount by time τ_f for the corresponding CfPP solution. The transportation cost of a CfPP solution is calculated by adding up the transportation costs of traversed edges in its giant-tour.

Search mechanism and improvement strategy: As previously mentioned, a prominent challenge of the BM algorithm is to identify a giant-tour that allows for a scheduling plan with the largest processed amount and lowest transportation cost. With this goal, two objectives with priority levels are handled in the tabu search framework as follows. While searching the solution space, the algorithm assigns priorities to each neighboring giant-tour. A neighboring giant-tour $\pi' \in \mathcal{N}(\pi)$, which is compared to the current solution π, belongs to one of the following four priority classes (sets) that are ordered from highest to lowest:

1. $\mathcal{N}_{\text{I}}(\pi)$ contains neighboring giant-tours that lead to CfPP solutions with larger processed amount compared to that of π.

2. $\mathcal{N}_{\text{II}}(\pi)$ contains neighboring giant-tours that lead to CfPP solutions with the same processed amount and smaller transportation cost compared to those of π.

3. $\mathcal{N}_{\text{III}}(\pi)$ contains neighboring giant-tours that lead to CfPP solutions with the same processed amount and the same or larger transportation cost compared to those of π.

4. $\mathcal{N}_{\text{IV}}(\pi)$ contains neighboring giant-tours that lead to CfPP solutions with a smaller processed amount compared to that of π.

With this notation, $\mathcal{N}(\pi) = \mathcal{N}_{\text{I}}(\pi) \bigcup \mathcal{N}_{\text{II}}(\pi) \bigcup \mathcal{N}_{\text{III}}(\pi) \bigcup \mathcal{N}_{\text{IV}}(\pi)$. While searching the neighborhood of a current giant-tour π, if a neighboring giant-tour that belongs to the priority class $\mathcal{N}_{\text{I}}(\pi)$ is encountered, then it is selected as the current giant-tour of the next iteration due to the higher priority of maximizing the processed amount by time τ_f. Since the whole neighborhood has not been scanned yet, this is a kind of *first improvement strategy*. If there is no neighboring giant-tour that belongs to the priority class $\mathcal{N}_{\text{I}}(\pi)$, then the neighboring giant-tour that belongs to $\mathcal{N}_{\text{II}}(\pi)$ with the smallest transportation cost is selected as the next giant-tour and is identified as π^*_{II}. In this case, since the whole neighborhood has been scanned, a *best improvement strategy* is taken. Therefore, a combination of the first and best improvement strategies is used by the BM algorithm. Note that, during the search of the neighborhood of π, the neighboring solutions, π^*_{II}, π^*_{III}, and π^*_{IV}, are updated.

Since the *first improvement strategy* is used by our BM algorithm, the order of scan of the neighborhoods of the move operations affects the best-found solution. After sample runs, we identified the best order of scan of the neighborhoods of the move operations as *Insert*, *Mutate*, *Delete*, and *Swap*. The impact of the selected improvement strategy and the order of scan of the neighborhoods of the move operations are examined in Section 6.2.3 through experimental tests.

Diversification: Diversification techniques are often used in tabu search algorithms to escape from local optima. We consider the following two strategies for this purpose.

1. When both $\mathcal{N}_{\text{I}}(\pi)$ and $\mathcal{N}_{\text{II}}(\pi)$ are empty, then the neighboring giant-tour, identified as $\pi^*_{\text{III}} \in \mathcal{N}_{\text{III}}(\pi)$, which has the smallest transportation cost in $\mathcal{N}_{\text{III}}(\pi)$, is selected as the giant-tour of next iteration.

2. If $\mathcal{N}_{\mathrm{III}}(\pi)$ is also empty in addition to $\mathcal{N}_{\mathrm{I}}(\pi)$ and $\mathcal{N}_{\mathrm{II}}(\pi)$, then the neighboring giant-tour, identified as $\pi^*_{\mathrm{IV}} \in \mathcal{N}_{\mathrm{IV}}(\pi)$, which has the largest processed amount in $\mathcal{N}_{\mathrm{IV}}(\pi)$, is selected as the giant-tour of next iteration.

Termination: We use two termination criteria in the algorithm. The algorithm stops after a chosen number of iterations (η) or a specified number of iterations without an improvement (γ) in the objective function.

6.1.5 *The algorithm*

Algorithm 2 The BM algorithm

Construct the initial solution g_0, which has a giant-tour π_0.
Set $\pi \leftarrow \pi_0$, $\pi^* \leftarrow \pi_0$ and calculate $P^{\pi^*}(\tau_f)$ and $C^{\pi^*}(\tau_f)$.
Initialize *Tabu Lists*.
while termination criteria not satisfied **do**
  Set $P(\tau_f) = 0$ and $C(\tau_f) = \infty$ for $\pi^*_{\text{II}}, \pi^*_{\text{III}}, \pi^*_{\text{IV}}$.
  for $\pi' \in \mathcal{N}(\pi)$ **do**
    if $P^{\pi'}(\tau_f) > P^{\pi^*}(\tau_f)$ **then** // π' improves the best found $P(\tau_f)$
      Set $\pi \leftarrow \pi'$.
      Update π^*, $P^{\pi^*}(\tau_f)$ and $C^{\pi^*}(\tau_f)$. // First improvement strategy
      go to **UPDATE**.
    else if $P^{\pi'}(\tau_f) = P^{\pi^*}(\tau_f)$ **then**
      if $C^{\pi'}(\tau_f) < C^{\pi}(\tau_f)$ **then** // π' improves the best found $C(\tau_f)$
        if $C^{\pi'}(\tau_f) < C^{\pi^*_{\text{II}}}(\tau_f)$ **then**
          Set $\pi^*_{\text{II}} \leftarrow \pi'$. // Update the candidate solution
        end if
      else if $C^{\pi'}(\tau_f) < C^{\pi^*_{\text{III}}}(\tau_f)$ **then** // No improvement
        Set $\pi^*_{\text{III}} \leftarrow \pi'$. // Update the candidate diversifying solution
      end if
    else if $P^{\pi'}(\tau_f) > P^{\pi^*_{\text{IV}}}(\tau_f)$ **then** // No improvement
      Set $\pi^*_{\text{IV}} \leftarrow \pi'$. // Update the candidate diversifying solution
    end if
  end for

Algorithm 2 The BM algorithm(cont.)

```
    if π*_II ≠ NULL then
        Set π ← π*_II.
        Update π*, P^{π*}(τ_f) and C^{π*}(τ_f). // Best improvement strategy
    else if π*_III ≠ NULL then
        Set π ← π*_III. // Diversify
    else
        Set π ← π*_IV. // Diversify
    end if
    UPDATE: Update the Tabu List of the move operation used to obtain π.
end while
```

6.2 Computational experiments

In this section, we analyze the performance of the solutions obtained by the proposed solution approach using numerical experiments with two groups of test instances. The first group is composed of the benchmark data sets introduced in Chapter 5 that has a number of nodes varying between 10 and 18, and the second group includes a new set of data instances tailored from the selected CVRP (Capacitated Vehicle Routing Problem) instances from the literature with up to 51 nodes. In this section, we explain how the new test problems were generated, and the results obtained with the algorithm on both groups of test problems. We compare the solution values of the BM algorithm with those of TF heuristic proposed in Chapter 5. In addition, we evaluate the performance of solutions obtained from the BM algorithm using the upper bound on the processed amount by time τ_f and the lower bound on the transportation cost, which are derived in Chapter 4. Since the TBH algorithm is used as an initial solution generator for the BM, through the computational analysis we also aim to see how much the BM algorithm improves the solution of the TBH in terms of both objectives.

The BM approach was implemented in C++. ILOG CPLEX 12.2 was used to solve the LP model of the BM algorithm. The computational experiments were performed on a Xeon E5520 @ 2.27 GHz processing unit with 48GB of memory.

6.2.1 Description of the test instances

We apply our BM approach on two groups of instances. The first group is composed of test instances, which are defined in Chapter 5 based on the data of a clinical laboratory, and include 12 instance sets, where each set is defined on a different complete graph, $G = (N^+, E)$ with the number of nodes in N^+ varying between 10 and 18. In each instance set, there are three instances generated according to three workload levels ($\alpha_2 = 0.8, 1.0, 1.2$). As a result, there are 36 data instances in total in the first group.

In the second group, we tailored a set of CVRP data instances from the VRPLIB-DEIS. We selected 3 different problem sizes from VRPLIB-DEIS (31, 41, and 51), each of which corresponds to an instance set. In the following paragraphs, we describe how the CVRP instances are tailored to obtain CfPP instances.

All nodes except the depot have demands in a CVRP instance. By default, we take the demand of a node $i \in N$ in a CVRP instance as the accumulation rate of the site, λ_i, in the corresponding CfPP instance.

Similar to the first group of CfPP instances, in each instance set, there are three instances generated according to three workload levels ($\alpha_2 = 0.8, 1.0, 1.2$) by varying μ. The value of μ for the corresponding instance is defined using the equality $\tau_e \sum_i \lambda_i = \alpha_2 \mu \tau_f$. As a result, there are 9 data instances in total in the second group of CfPP instances.

Distance matrices, d_{ij}'s, of the CVRP instances remain as given in the corresponding CfPP instances and the travel time of edge (i, j) is assumed to be equal to the traveling distance, i.e., $t_{ij} = d_{ij}$.

In the first group of instances $\tau_e = 660$, and $\tau_f = 1,260$ minutes. This setting remains the same in the second group.

The data sets can be accessed at http://home.ku.edu.tr/eyucel/Research/Dataset CfPP.zip.

6.2.2 Experimental setting

The BM algorithm involves five parameters: $(\eta, \gamma, \theta, \kappa_1, \kappa_2)$ (see Table 6.3). On the basis of preliminary tests, we set these parameter values as $\eta = 500, \gamma = 6, \theta = 8, \kappa_1 = 10, \kappa_2 = n$.

Because of the randomness associated with the *Mutate* move operation, different runs of the BM algorithm lead to possibly different solutions. We made some preliminary tests with different seeds of the random numbers generator, and we observed that 5 experimental runs are sufficient to evaluate the quality of the BM. Thus, we generated 5 experimental runs in the following tests. From these runs, the BM solution with the best processed amount is provided. The total run time of 5 runs (in seconds) is reported for the BM algorithm. The TF heuristic in Chapter 5 was run for a total computation time of six hours (21600 s) on the same computation platform that the experiments of the BM were done. Note that the running time of the TBH algorithm is approximately 1 s, which is negligible.

6.2.3 Experimental results

In this section, we first compare the solutions found by the BM, TF, and TBH algorithms in terms of the processed amount by time τ_f and the transportation cost for both groups of CfPP test instances. We report the solutions with respect to the deviations from

$\mathrm{UB}_{P(\tau_f)}$ the upper bound presented in Chapter 4 on the processed amount by time τ_f that is found by relaxing the single vehicle restriction and assuming that there are infinitely many vehicles, and

$\mathrm{LB}_{C(\tau_f)}$ the best lower bound on the transportation cost obtained from the two relaxations given in Chapter 4, which are defined for a given level of the processed amount, at the end of a run time limit of four hours (14400 s).

In addition, we analyze the impact of some characteristics of the BM algorithm on the solution quality. Finally, we investigate the effect of the problem size and workload level on the solutions and the problem difficulty.

The comparative results of the BM, TF, and TBH algorithms on the first and second group of CfPP test instances are provided in Table 6.6 and Table 6.7, respectively. In the tables, the first three columns specify the instance. Then, for each algorithm, three columns report the processed amount by time τ_f ($P(\tau_f)$), the transportation cost ($C(\tau_f)$), and the percentage gap between $P(\tau_f)$ and $\mathrm{UB}_{P(\tau_f)}$, which is referred to as $\mathrm{Gap}_{P(\tau_f)}$ and calculated as $100(\mathrm{UB}_{P(\tau_f)} - P(\tau_f))/P(\tau_f)$. For the BM solutions, we also report the run time under the CPU sec column. For each instance, for the solution with the best found $P(\tau_f)$ value among the BM and TF solutions, the percentage gap between $C(\tau_f)$ and $\mathrm{LB}_{C(\tau_f)}$, which is referred to as $\mathrm{Gap}_{C(\tau_f)}$ and calculated as $100(C(\tau_f) - \mathrm{LB}_{C(\tau_f)})/(\mathrm{LB}_{C(\tau_f)})$, is reported in parenthesis besides $C(\tau_f)$ value. In Table 6.7, rows with * sign indicate that the TF approach cannot find a feasible solution at the end of the given run time limit.

Instance			BM				TF			TBH		
Set	n	α_2	$P(\tau_f)$	$C(\tau_f)$	CPU sec	$\text{Gap}_{P(\tau_f)}(\%)$	$P(\tau_f)$	$C(\tau_f)$	$\text{Gap}_{P(\tau_f)}(\%)$	$P(\tau_f)$	$C(\tau_f)$	$\text{Gap}_{P(\tau_f)}(\%)$
		0.8	10493	160 (0.00)	427	0.00	10493	171	0.00	10493	741	0.00
1	14	1	10185	321 (23.46)	1498	1.23	10185	321	1.23	10080	716	2.28
		1.2	8507	280 (21.21)	1682	1.07	8507	284	1.07	8384	672	2.56
		0.8	4786	135 (4.65)	61	0.00	4786	143	0.00	4786	649	0.00
2	9	1	4658	328 (20.59)	334	1.21	4658	328	1.21	4544	493	3.74
		1.2	3886	260 (21.50)	275	1.16	3886	260	1.16	3858	497	1.90
		0.8	4522	168 (4.35)	218	0.00	4522	173	0.00	4522	739	0.00
3	12	1	4389	352 (27.54)	934	1.27	4389	352	1.27	4308	637	3.16
		1.2	3665	302 (23.27)	1079	1.13	3665	306	1.13	3592	559	3.18
		0.8	4509	203 (4.64)	462	0.00	4509	218	0.00	4509	805	0.00
4	13	1	4361	415 (26.52)	1956	1.56	4361	415	1.56	4283	701	3.41
		1.2	3651	362 (25.69)	2417	1.16	3651	362	1.16	3570	640	3.44
		0.8	4540	211 (2.93)	456	0.00	4540	215	0.00	4540	848	0.00
5	14	1	4385	457 (27.65)	979	1.75	4362	379	2.30	4282	833	4.19
		1.2	3668	350 (21.11)	2493	1.38	3668	351	1.38	3633	804	2.36
		0.8	4928	169 (2.42)	249	0.00	4928	169	0.00	4928	662	0.00
6	11	1	4746	334 (24.63)	722	1.93	4746	334	1.93	4697	618	2.98
		1.2	3983	313 (19.92)	854	1.23	3983	320	1.23	3915	559	3.00
		0.8	4807	164 (2.50)	152	0.00	4807	170	0.00	4807	640	0.00
7	11	1	4643	344 (25.55)	999	1.46	4643	344	1.46	4575	626	2.97
		1.2	3877	325 (20.82)	1076	1.28	3873	290	1.39	3812	542	3.00
		0.8	4455	169 (3.05)	205	0.00	4455	173	0.00	4455	702	0.00
8	12	1	4305	352 (27.08)	1206	1.64	4301	351	1.74	4184	550	4.59
		1.2	3601	312 (22.83)	1680	1.36	3601	312	1.36	3487	526	4.66

continued on next page

continued from previous page

Instance			BM				TF			TBH		
Set	n	α_2	$P(\tau_f)$	$C(\tau_f)$	CPU sec	$\text{Gap}_{P(\tau_f)}(\%)$	$P(\tau_f)$	$C(\tau_f)$	$\text{Gap}_{P(\tau_f)}(\%)$	$P(\tau_f)$	$C(\tau_f)$	$\text{Gap}_{P(\tau_f)}(\%)$
		0.8	4763	148 (2.07)	177	0.00	4763	150	0.00	4763	570	0.00
9	11	1	4614	321 (27.89)	933	1.48	4613	317	1.48	4501	518	4.00
		1.2	3854	274 (21.78)	1001	1.28	3849	270	1.42	3824	435	2.08
		0.8	5798	215 (1.42)	1270	0.00	5798	216	0.00	5798	773	0.00
10	16	1	5592	428 (22.29)	3000	1.92	5592	428	1.92	5442	654	4.72
		1.2	4673	380 (25.83)	4609	1.69	4666	378	1.85	4636	682	2.50
		0.8	6214	227 (0.89)	1398	0.00	6214	229	0.00	6214	767	0.00
11	17	1	5927	413 (25.15)	2196	3.23	5906	387	3.59	5750	771	6.40
		1.2	4968	365 (23.73)	5284	2.69	4967	357	2.71	4792	681	6.46
		0.8	4687	173 (1.17)	535	0.00	4687	173	0.00	4687	741	0.00
12	14	1	4525	361 (26.67)	1513	1.85	4502	311	2.38	4508	708	2.24
		1.2	3790	321 (24.42)	2823	1.41	3784	291	1.57	3757	591	2.30
Avg.					1340	1.04			1.1			2.28
Max.					5284	3.23			3.59			6.46

Table 6.6: Test results on CfPP test instances

The results prove the effectiveness of the BM algorithm on both groups of CfPP instances. In all CfPP instances, the BM solutions dominate the TF solutions in terms of both objectives. The percentage deviation on the processed amount between the TF and BM solutions is less than 1% on the average for the first group of CfPP instances. In Table 6.7, we see that as the problem size and the workload level increase, the TF heuristic cannot even find a feasible solution at the end of the given run time limit. For the instances in the second group that the TF can find a solution at the end of run time limit, the percentage deviation on the processed amount between the TF and BM solutions is 6% on the average for the ones. This shows the effectiveness of the BM algorithm on larger CfPP instances.

For the first group of CfPP instances that the BM finds solutions with strictly larger $P(\tau_f)$ value, the percentage deviation on the transportation cost between the TF and BM solutions is -7% on the average. On the other hand, for the second group of CfPP instances, the percentage deviation on the transportation cost between the TF and BM solutions is 20.1% on the average.

Next, we compare the BM and TF heuristics in terms of the computation time. The results reported in Tables 6.6 and 6.7 show that the BM is faster than the TF heuristic which has a total run time of 21600 s. According to Table 6.7, for larger CfPP instances with higher workload levels, the TF heuristic cannot even find a feasible solution at the end of this run time limit. On the other hand, it takes 18711 s on the average (maximum = 21502 s) for the BM algorithm to find the best solution.

Table 6.7: Test results on CfPP test instances tailored from CVRP instances

Instance		BM				TF			TBH		
n	α_2	$P(\tau_f)$	$C(\tau_f)$	CPU sec	$\text{Gap}_{P(\tau_f)}(\%)$	$P(\tau_f)$	$C(\tau_f)$	$\text{Gap}_{P(\tau_f)}(\%)$	$P(\tau_f)$	$C(\tau_f)$	$\text{Gap}_{P(\tau_f)}(\%)$
	0.8	389400	461 (8.47)	16445	0.00	377318	445	3.20	389400	1035	0.00
30	1	350151	566 (26.91)	13218	9.86	337166	744	14.09	322646	887	19.23
	1.2	296317	533 (26.90)	17557	8.25	287209	772	11.68	268871	886	19.30
	0.8	526680	500 (10.86)	18452	0.00	520092	728	1.27	526680	907	0.00
40	1	466050	462 (31.62)	19929	11.85	425376	580	22.54	453112	897	15.04
	1.2	398049	427 (31.38)	20340	9.20	*			377593	914	15.12
	0.8	512820	893 (18.12)	20507	0.00	453605	519	13.05	512820	979	0.00
50	1	473850	982 (29.38)	20451	7.29	*			467357	960	8.78
	1.2	404679	981 (32.39)	21502	4.73	*			400663	962	5.78
Avg.				18711	5.69			10.97			9.25
Max.				21502	11.85			22.54			19.30

From these results, we can also examine how much the initial solution found by the TBH heuristic improves through the BM algorithm. For the first group of instances, the initial solution is improved by the BM 1.2% on the processed amount and 56.5% on the transportation cost, on the average. For the second group of instances, the initial solution is improved by the BM 2.7% on the processed amount and 34.4% on the transportation cost, on the average.

Next, we analyze the impact of some characteristics of the BM algorithm on the solution quality.

Impact of the improvement strategy

Although a general approach used in tabu search applications is to use only best improvement strategy, our BM algorithm uses both first and best improvement strategy, together. In order to evaluate the impact of this novel improvement strategy on the quality of the solution generated by the BM algorithm, the first instance set in Table 6.6 is chosen. We implemented a variant of the BM algorithm that uses only best improvement strategy. The results shown in Table 6.8 refer to the best found solutions of the original BM algorithm and its variant (the BM without first improvement strategy). In the table, $P'(\tau_f)$ and $C'(\tau_f)$ stand for the processed amount and the transportation cost of the best found solution of the BM variant. From the table, it can be observed that the original BM algorithm finds dominating solutions in terms of both objectives in a shorter amount of time than its variant. Clearly, the reason for this is that if first improvement strategy is not applied, then it requires more time to pass to the next iteration on the average. When $\alpha_2 = 0.8$, both algorithms find the same solution, but the computation time of the BM variant is longer. Figure 6.1 shows how both objective values change on time for $\alpha_2 = 1$ and 1.2 for both of the algorithms. The figure shows that for the selected instance set the first improvement strategy diversifies the search.

Table 6.8: Test results for the selected instance set provided by the BM with and without first improvement strategy

Instance		BM			BM without first improvement strategy		
Set	α_2	$P(\tau_f)$	$C(\tau_f)$	CPU sec	$P'(\tau_f)$	$C'(\tau_f)$	CPU sec
	0.8	10493	160	427	10493	160	863
1	1	10185	329	1498	10171	334	2006
	1.2	8507	280	1682	8471	288	1944
Avg.		9346	304.5	1590	9321	311	1975

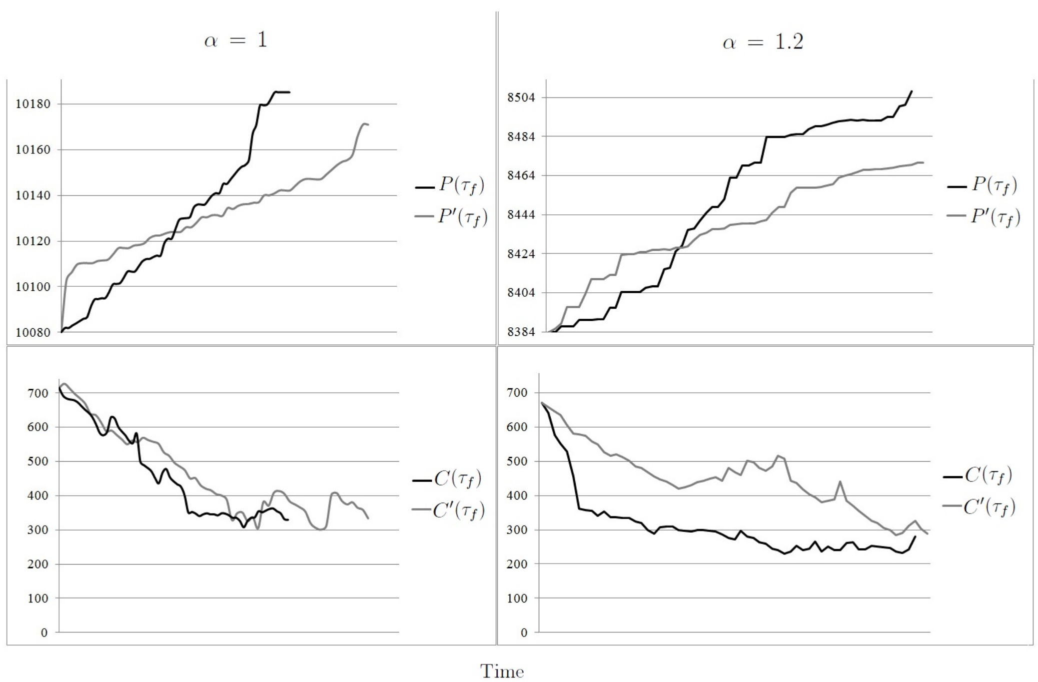


Figure 6.1: Change in the objective values on time for $\alpha_2 = 1$ and 1.2 for BM with and without first improvement strategy

Impact of setting κ_1 in a dynamic way

Note that our BM algorithm constructs solutions that have at most κ_1 tours before τ_e and κ_1 is initially set to 10 during the computational experiments. However, if the TBH solution has more than κ_1 tours before τ_e, then the BM increases κ_1 to the number of D-tours of the initial solution generated by the TBH. Now, we test the impact of setting κ_1 in such a dynamic way inside the BM algorithm. For this purpose, we implement the variant of the algorithm with fixed κ_1, i.e., $\kappa_1 = 10$. Since the number of D-tours of the initial solution generated by the TBH is larger than 10 for the instance set with size 51 in Table 6.7, this instance set was chosen for these tests. The results shown in Table 6.9 provide the solutions of the original algorithm and the variant with fixed κ_1. In the table, for both the original BM and its variant, five columns report the processed amount by time τ_f ($P(\tau_f)$), the transportation cost ($C(\tau_f)$), the number of D-tours (K_1), the number of A-tours (K_2), and the percentage of visited sites in each D-tour (V_1) of the best found solutions. The last column in the table, $\text{Gap}^{\kappa_1}_{P(\tau_f)}$, reports the percentage gap between the best found processed amounts. The table shows that the gap between the processed amounts of the best found solutions is 7.3% on the average. The number of D-tours of the best found solutions of the original BM are nearly three times larger than those of the BM with fixed κ_1, on the average. However, the percentage of sites visited in each tour of the original BM solutions are nearly half of those of the BM variant, on the average, meaning that setting κ_1 dynamically results in solutions that perform frequent and short tours to feed the processor at the processing facility.

Table 6.9: Test results for the selected instance set provided by the BM with and without fixed $\kappa_1 = 10$

Instance		BM					BM with fixed $\kappa_1 = 10$					$\mathrm{Gap}^{\kappa_1}_{P(\tau_f)}$
n	α_2	$P(\tau_f)$	$C(\tau_f)$	K_1	K_2	$\overline{v}_1$	$P(\tau_f)$	$C(\tau_f)$	K_1	K_2	$\overline{v}_1$	(%)
	0.8	512820	893	24	1	12.8	467986	705	6	1	23.3	9.6
50	1	473850	982	19	1	14.6	439584	644	7	1	33.7	7.8
	1.2	404679	981	18	1	18.1	386674	628	7	1	32.7	4.5
Avg.		463783	952	20.3	1	15.2	431415	659	6.7	1	29.9	7.3

Impact of the order of scan of the neighborhoods of the move operations

Since the *first improvement strategy* is used in our BM algorithm, the order of scan of the neighborhoods of the move operations affects the best-found solution. Note that the original order of scan of neighborhoods in our BM algorithm is *Insert*(I)-*Mutate* (M)-*Delete*(D)-*Swap*(S), which we refer to as I-M-D-S in short. Now, we test the BM algorithm with different order of scan of the neighborhoods of the move operations. We again picked the first instance set in Table 6.6 to perform these tests. Since there are four types of move operations, there are 24 different possible orderings. Although we tried all of the possible orderings in our sample runs, due to space limitations, in Table 6.10, we report the results for the BM variants with one of the three selected orderings (M-I-D-S, I-M-S-D, I-D-M-S), which provide better results compared to the others, in addition to the original BM with ordering of I-M-D-S. In the table, for each BM variant, two columns report the processed amount by time τ_f ($P(\tau_f)$) and the transportation cost ($C(\tau_f)$) of the best found solution. According to the table, the best found solutions of the original BM algorithm dominate the others although the BM variants with the orders M-I-D-S and I-D-M-S find close solutions.

Table 6.10: Test results for the selected instance set with different orders of neighborhood scan

Instance		BM		BM with M-I-D-S		BM with I-M-S-D		BM with I-D-M-S	
Set	α_2	$P(\tau_f)$	$C(\tau_f)$	$P(\tau_f)$	$C(\tau_f)$	$P(\tau_f)$	$C(\tau_f)$	$P(\tau_f)$	$C(\tau_f)$
	0.8	10493	160	10493	162	10493	181	10493	160
1	1	10185	329	10185	329	10179	354	10185	348
	1.2	8507	280	8498	277	8467	293	8507	282

Impact of the problem size

Now, we analyze the impact of the problem size on the problem difficulty and the quality of solutions found by our BM algorithm. Figure 6.2 provides a bar chart extracted from Tables 6.6 and 6.7 on the statistics of $\text{Gap}_{P(\tau_f)}$ value for different problem sizes. According to the figure, although the best found solutions have a processed amount very close to that of the infinite vehicle relaxation, for larger problem sizes the maximum gap increases up to 11.85%. These relatively large gaps may be attributed to several factors: (1) the weakness of the upper bounds, (2) the possibility that the best found solutions may be far from the optimal, (3) inherent computational difficulty of the problem for large problem sizes.

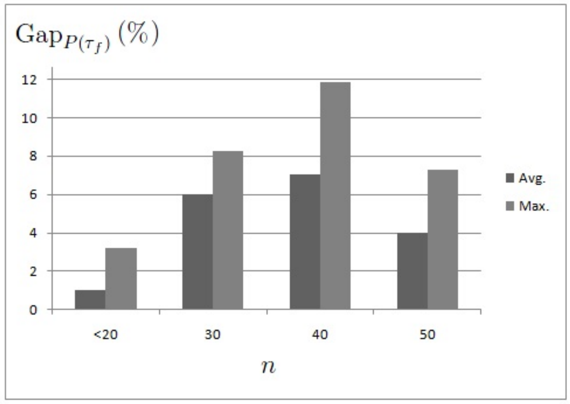


Figure 6.2: $\mathrm{Gap}_{P(\tau_f)}$ statistics of the BM solutions for different problem sizes

Impact of the workload level

Next, we analyze the effect of workload level on the transportation cost and the number of tours. Table 6.11 and Table 6.12 report the statistics on the transportation costs and the average number of tours for different workload levels for the first and second groups of instances, respectively. In the tables, the results of the BM and TF solutions are reported separately. According to Table 6.11, similar to the TF solutions, in the BM solutions, the vehicle performs fewer tours when $\alpha_2 = 0.8$ paying the lowest transportation cost on the average. Both approaches incur the highest cost and the highest number of tours on the average when $\alpha_2 = 1$. Here, we note that as stated in previous paragraphs the BM solutions outperform the TF solutions in terms of the processed amount, therefore Tables 6.11 and 6.12 cannot be used to compare the BM and TF solutions in terms of the transportation cost and the number of tours.

Table 6.11: Average results over each workload level for the first group of instances

α_2		Avg. $P(\tau_f)$	Avg. $C(\tau_f)$	Avg. K_1	Avg. K_2	Avg. $\overline{v}_1$ (%)	Avg. $\text{Gap}_{P(\tau_f)}$ (%)
0.8	BM	5375	178.5	2.8	1.2	15.9	0
	TF	5375	183	3.50	1	15.5	0
1	BM	5194	369.5	8.3	1	18.9	1.7
	TF	5188	356	8.92	1	15.6	1.84
1.2	BM	4344	320.3	7.4	1	17.2	1.4
	TF	4342	315	7.75	1	15.15	1.45

Table 6.12: Average results over each workload level for the second group of instances

α_2		Avg. $P(\tau_f)$	Avg. $C(\tau_f)$	Avg. K_1	Avg. K_2	Avg. $\overline{v}_1$ (%)	Avg. $\text{Gap}_{P(\tau_f)}$ (%)
0.8	BM	476300	618	10.3	1	29.9	0
	TF	450338	564	4.33	1	26.4	5.84
1				not valid			
1.2				not valid			

Alternatives to prioritized objectives

As stated in Section 4.2.1, there are alternative approaches to deal with MOOPs with a priori prioritized objectives. Now, we consider two other approaches to handle the two objectives of CfPP and evaluate their suitability for CfPP optimization. First, the common approach of taking a weighted sum of the objectives is considered and a composite single objective function is generated from the two objectives of CfPP as $\omega_1 P(\tau_f) + \omega_2 C(\tau_f)$, where $\omega_1 + \omega_2 = 1$. To analyze the effect of this approach on the solution quality, we implemented a variant of the BM algorithm that maximizes only this composite single objective function. We call this variant of BM as $M^{<\omega_1,\omega_2>}$. During our experimental analysis, we considered two sets for $< \omega_1, \omega_2 >$ as $< 0.7, 0.3 >$ and $< 0.6, 0.4 >$.

Second, as another way of unifying the two objectives, the performance metric, CPTU, is defined as the cost paid per 1,000 units processed, and we implemented another variant of the BM algorithm that minimizes only CPTU. We call this variant of BM as M^{CPTU}.

Again, we picked the first instance set in Table 6.6 for these tests. In Table 6.13, we report the best found solutions of the BM variants. In the table, for each of the variants, three columns report the processed amount by time τ_f ($P(\tau_f)$), the transportation cost ($C(\tau_f)$), and the CPTU value of the best found solution. According to the results, the original BM finds better results in terms of both objectives for this instance set.

Table 6.13: Test results for the selected instance set with different objectives

Instance		BM			$M^{<0.6,0.4>}$			$M^{<0.7,0.3>}$			M^{CPTU}		
Set	α_2	$P(\tau_f)$	$C(\tau_f)$	CPTU	$P(\tau_f)$	$C(\tau_f)$	CPTU	$P(\tau_f)$	$C(\tau_f)$	CPTU	$P(\tau_f)$	$C(\tau_f)$	CPTU
	0.8	10493	160	15.2	10493	171	16.3	10493	181	17.2	10493	160	15.2
1	1	10185	329	32.3	10149	311	30.6	10179	354	34.8	10153	307	30.2
	1.2	8507	280	32.9	8500	243	28.6	8484	231	27.2	8500	233	27.4

6.3 Conclusions and summary of contributions

In this chapter, we present a bicriteria matheuristic, BM, to address the single vehicle collection for processing problem with two prioritized objectives. The proposed approach combines a tabu search scheme with exact LP models in a way that the use of LP models speeds up the search process by means of completing the partial solution of tabu search optimally. Our heuristic handles the two prioritized objectives of the problem through a novel approach by classifying the neighborhood into priority classes and using a combination of first and best improvement strategies to update the current solution. Priority classes also enhance the diversification.

Our heuristic has been tested on realistic data sets. The results show that the proposed heuristic outperforms the TF heuristic developed in Chapter 5 by providing effective solutions for large-size CfPP instances within reasonable computation time. Through numerical tests we also show that the novel features of our algorithm are largely responsible for its effectiveness in the computational experiments.

Chapter 7

CONCLUSION

With developments in medical technologies and increasing use of clinical testing procedures, clinical testing laboratories are likely to continue their crucial role in the delivery of healthcare services. This thesis has addressed the specimen collection problem that clinical testing laboratories face daily, and often results in lost processing capacity, penalties due to late delivery of test results, increased processing and transportation costs, and potential negative impact on patient health. The problem was formalized as collection for processing problem (CfPP). Specifically, CfPP aims to transport items that accumulate at a number of geographically-dispersed sites throughout the working day to a centralized facility for subsequent processing on equipment with limited capacity.

We studied two variants of the collection for processing problem in this thesis: (i) multi-vehicle problem under the objective of minimizing total completion time of all the accumulated amount, i.e., makespan, which is defined as mCfPP($C_{\max}$), and (ii) single vehicle problem under the objectives of maximizing the total processed amount as a first priority and minimizing the transportation costs as a second priority.

We proposed the first approximation result for mCfPP($C_{\max}$). We proved that the problem is NP-hard by a reduction from a two-stage, hybrid flowshop scheduling problem. We analyzed the special case with a single site to find the number of vehicles necessary to achieve the minimum makespan and to identify the minimum makespan for a single vehicle. Using the insights obtained from these results, we developed a clustering-based, constant-factor approximation algorithm for this strongly NP-hard problem. The proposed approximation algorithm is based on the idea of partitioning the sites into clusters, and solving a single vehicle problem for each cluster. Since assigning a single vehicle to a dedicated region of sites is a practically used approach in the real-world problem and the worst-case bound of the proposed algorithm is motivating, we studied the single vehicle problem.

We studied the single vehicle problem with two prioritized objectives. We showed that this problem is NP-hard and formulated a mixed integer programming (MIP) model for its solution. We observed that since the solution space is extremely large even for small problem sizes, the MIP model strengthened with valid inequalities cannot find good solutions within reasonable time. Therefore, we proposed two heuristic approaches to address the single vehicle problem. We also presented a procedure to calculate an upper bound on the processed amount by a deadline and two relaxed MIP models to generate lower bounds on the transportation cost.

The first heuristic for the single vehicle problem incorporates additional constraints to the MIP model with the goal of eliminating solutions that are likely to be suboptimal. We showed through computational tests that the proposed approach finds solutions that are very close to optimal in terms of the processed amount. Furthermore, we showed that transportation costs of these solutions are in the vicinity of 13% to optimality on the average. In addition, we identified the workload level and the pattern of the accumulation at the sites as key indicator parameters for problem difficulty.

In order to solve larger instances of the single vehicle problem, we presented a bicriteria matheuristic, that combines a tabu search scheme with exact linear programming models. Numerical experiments conducted on the realistic data sets showed that the proposed heuristic was able to provide effective solutions for actual large-size instances within reasonable computation time. In the proposed bicriteria matheuristic, the use of linear programming models enhanced the search process by means of completing the partial solution of tabu search optimally. By classifying the neighborhood into priority classes and using a combination of first and best improvement strategies to update the current solution, the heuristic handled two objectives with priority levels while searching the solution space. Priority classes also enhanced the diversification. These features were largely responsible for the effectiveness of the proposed approach in the computational experiments.

In conclusion, the proposed research not only yields practical insights and heuristics, but also significant extensions to the theory through novel solution methodologies for the generalized problem of coordinating transportation decisions and subsequent processing operations. Hence, while the problem is described in the context of a healthcare logistics problem, we expect that the results obtained in this thesis would be useful to the logistics problems encountered in environments with accumulation of items at the nodes and subsequent processing of these items at a centralized facility where processing is a critical concern.

7.1 Future research directions

There are the following ample opportunities for further research in the context of the collection problem in clinical laboratories.

An important characteristic in the real-world problems is that accumulation rates at the sites change throughout the day; typically the specimen accumulation rates are higher in the morning than the other times of the day, but the delivery of the specimens at the processing center are later in the afternoons. The pickups and delivery need to be coordinated so that the expensive processing equipment is better utilized. In addition to making better decisions regarding the collection of items from the various locations, the processing rate of the equipment can be adjusted throughout the day to better match the collection rate. Explicit modeling of the nonstationary nature of accumulation at the nodes, in conjunction with the consideration of possibly changing the processing capacity throughout the day is likely to yield significant improvements in the amount of specimens that can be processed per day, and as well as reductions in the labor, operating, and transportation costs.

While processing equipment requires significant capital investment, it is conceivable that processing is performed in multiple locations rather than a single centralized location. A promising research direction is to investigate the implications of extending our models to the case of multiple processing facilities, so that selection of the processing center is considered in addition to the routing/scheduling of the visits.

It would also be interesting to consider information/data availability at the collection sites as an additional problem characteristic. If the status of each collection site is available in real-time, then dispatching vehicles and scheduling of collection times can be made more efficient.

Finally, another interesting topic for further research would be to address the problem of locating satellite pre-processing facilities to decrease delays at the central facility – where and with what capacity. Candidate locations for these units are the laboratory's own patient service centers. In particular, the various collection sites need to be assigned to the various pre-processing locations, and these locations have to be staffed appropriately. The research on this problem might be broadened to answer how the specimens should then be transported to the central facility.

BIBLIOGRAPHY

J.H. Ahmadi, R.H. Ahmadi, S. Dasu, and C.S. Tang. Batching and scheduling jobs on batch and discrete processors. *Operations Research*, 40:750–763, 1992.

F. Alonso, M.J. Alvarez, and J.E. Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59:963–976, 2008.

D. Applegate, W. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing*, 14:132143, 2002.

C. Archetti, A. Hertz, and M. Speranza. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40:64–73, 2006.

C. Archetti, A. Hertz, and M. Speranza. Metaheuristics for the team orienteering problem. *J Heuristics*, 13:46–76, 2007.

E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34:209–219, 2006.

F. Ben Abdelaziz, S. Krichen, and J. Chaouachi. A hybrid heuristic for multiobjective knapsack problems. In I.H. Osman S. Voss, S. Martello and C.Roucairol, editors, *Meta Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer, 1999.

N. Bianchessi and G. Righini. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers and Operations Research*, 34: 578–594, 2006.

M.A. Boschetti, V. Maniezzo, M. Roffilli, and A.B. Röhler. Matheuristics: Optimization, simulation and control. *Lecture Notes in Computer Science*, 5818:171–177, 2009.

S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5:211–230, 2007.

J. Brandao and A. Mercer. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100:180–191, 1997.

J. Brandao and A. Mercer. The multi-trip vehicle routing problem. *Journal of the Operational Research Society*, 49:799–805, 1998.

O. Braysy and M. Gendreau. Tabu search heuristics for the vehicle routing problem with time windows. *TOP*, 10:217–237, 2002.

O. Braysy and M. Gendreau. Vehicle routing problem with time windows, part i: route construction and local search algorithms. *Transportation Science*, 39:104–118, 2005a.

O. Braysy and M. Gendreau. Vehicle routing problem with time windows, part ii: metaheuristics. *Transportation Science*, 39:119–139, 2005b.

R. J. III Brideau and T. M. Cavalier. The maximum collection problem with time dependent rewards. TIMS Internat. Conf., Anchorage, AK, 1994.

S. Butt and D. Ryan. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers and Operations Research*, 26:427–441, 1999.

R. Caballero, M. Gonzalez, F.M. Guerrero, J. Molina, and C. Paralera. Solving a multiobjective location routing problem with a metaheuristic based on tabu search. Application to a real case in Andalusia. *European Journal of Operational Research*, 177:1751–1763, 2007.

A. M. Campbell, M. W. P. Savelsbergh, L. Clarke, and A. Kleywegt. Efficient insertion heuristics for vehicle routing and scheduling problems. In G. Laporte T. G. Crainic, editor, *The inventory routing problem.Fleet Management and Logistics*, pages 95–112. Kluwer Academic Publishers, Norwell, MA, 1998.

I. M. Chao, B.L. Golden, and E.A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88:475–489, 1996.

T.C.E. Cheng and G. Wang. Batching and scheduling to minimize the makespan in the two-machine flowshop. *IIE Transactions*, 30:447–453, 1998.

W. Chiang and R.A. Russell. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9:417–430, 1997.

N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical report, Technical report 338, Graduate School of Industrial Administration, CMU, 1976.

G. Clarke and J.V. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

J. F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo. Transportation, handbooks in operations research and management science. In G. Laporte C. Barnhart, editor, *Vehicle routing*, volume 14, pages 364–428. Elsevier, Amsterdam, 2007.

J.F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis. *The vehicle routing problem*, chapter Vrp with time windows, pages 157–193. SIAM Monographs on Discrete Mathematics and Applications. SIAM Publishing, 2002.

J.F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of Operations Research Society*, 52: 928–936, 2001.

M. DellAmico, F. Maffioli, and P. Vrbrand. On prize-collecting tours and the asymmetric travelling salesman problem. *Internat. Trans. Oper. Res*, 2:297–308, 1995.

K.F. Doerner, M. Gronalt, R.F. Hartl, G. Kiechle, and M. Reimann. Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. *Computers and Operations Research*, 35:3034–3048, 2008.

G. Dueck and T. Scheuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90:161–175, 1990.

M.B. Dumas and M. Rabinowitz. Policies for reducing boood wastage in hospital blood banks. *Management Science*, 23:1124–1132, 1977.

A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. *Journal of Algorithms*, 48:135–159, 2003.

M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 12:1–63, 2004.

D. Feillet, P. Dejax, and M. Gendreau. Travelling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

B. Fleischmann. The vehicle routing problem with multiple use of vehicles. Working Paper, Fachbereich Wirtschaftswissenschaften, Universitat Hamburg, 1990.

P.M. Franca, M. Gendreau, G. Laporte, and F.M. Mller. The m-traveling salesman problem with minmax objective. *Transportation Science*, 29:267–275, 1995.

G.N. Frederickson, M.S. Hecht, and C.E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7:178–193, 1978.

X. Gandibleux and A. Freville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objectives case. *Journal of Heuristics*, 6:361–383, 2000.

B. Gavish and S. C. Graves. The travelling salesman problem and related problems. Working Paper GR-078-78, Operations Research Center, Massachusetts Institute of Technology, 1978.

M. Gendreau, G. Laporte, and J. Y. Potvin. *The Vehicle Routing Problem*, chapter Metaheuristics for the capacitated VRP, pages 129–154. SIAM Monographs on Discrete Mathematics and Applications. SIAM Publishing, 2002.

M. Gendreau, G. Laporte, and R. Seguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44 (3):469–477, 1996.

G. Glover and G.A. Kochenberger. *Handbook of metaheuristics*. Kluwer, Boston, MA, 2003.

B. L. Golden, G. Laporte, and E. Taillard. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers and Operations Research*, 24(5):445–452, 1996.

B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.

Y. Y. Haimes, L.S. Lasdon, and D.A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetic*, 1(3):296–297, 1971.

T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno, and M. Yagiura. Effective local search algorithms for routing and scheduling problems with general time window constraints. *Transportation Science*, 39:206–232, 2005.

B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers and Operations Research*, 35:2307–2330, 2008.

I.Y. Kim and O.L. de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Struct Multidisc Optim*, 29:149–158, 2005.

Knowledge Source Inc., 2008. *Clinical Laboratory Testing Market Overview 2008*.

Koncept Analytics, 2008. *Worldwide Clinical Laboratory Testing Market*.

Y.A. Koskosidis, W.B. Powell, and M.M. Solomon. An optimization based heuristic for vehicle routing and scheduling with soft time window constraints. *Transportation Science*, 26:69–85, 1992.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. 21:498–516, 1973.

V. Maniezzo, T. Stützle, and S. Vo. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer, Berlin, 2009.

J. J. McDonald. Vehicle scheduling - a case study. *Operational Research Quarterly (1970-1977)*, 23:433–444, 1972.

N. H. Moin and S. Salhi. Inventory routing problems: a logistical overview. *Journal of the Operational Research Society*, 58:1185–1194, 2007.

J. W. Ohlmann, M.J. Fry, and B. W. Thomas. Route design for lean production systems. *Transportation Science*, 42(3):352–370, 2008.

A. Olivera and O. Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers and Operations Research*, 34:28–47, 2007.

I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Evanston, IL: Northwestern University, 1976.

R.J. Petch and S. Salhi. A multi-phase constructive heuristic for the vehicle routing problems with multiple trips. *Discrete Applied Mathematics*, 133:69–92, 2004.

L. Revere. Re-engineering proves effective for reducing courier costs. *Business Process Management Journal*, 10:400–414, 2004.

R. A. Russell and D. Gribbin. A multiphase approach to the period routing problem. *Networks*, 21:747–765, 7 1991.

S. Salhi and R.J. Petch. A ga based heurisric for the vehicle routing problem with multiple trips. *Journal of Mathematical Modeling and Algorithms*, 6:591–613, 2007.

F. Streichert and M. Tanaka-Yamawaki. A new scheme for interactive multi-criteria decision making. *Lecture Notes in Computer Science*, 4253:655–662, 2006.

E. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J. Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186, 1997.

E. D. Taillard, G. Laporte, and M. Gendreau. Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society*, 47:1065–1070, 1996.

H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers and Operations Research*, 32:1379–1407, 2005.

H. Tang, E. Miller-Hooks, and R. Tomastik. Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E*, 43:591–609, 2007.

VRPLIB-DEIS. *Library of Capacitated Vehicle Routing problem instances of D.E.I.S. Operations Research Group*. Università di Bologna. URL `http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html`. maintained by Daniele Vigo.

P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. Metaheuristics for tourist trip planning. In *Metaheuristics in the Service Industry*, Lecture Notes in Economics and Mathematical Systems, pages 15–31. Springer Berlin Heidelberg, 2009.

E. Weber, A. Rizzoli, R. Soncini-Sessa, and A. Castelletti. Lexicographic optimization for water resources planning: The case of lake Verbano, Italy. In A. Jakeman A. Rizzoli, editor, *Integrated Assessment and Decision Support Proceedings of the First Biennial Meeting of the the International Environmental Modelling and Software Society*, pages 235–240, Lugano, June 2002.

D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2010.

G. J. Woeginger. Exact algorithms for np-hard problems: A survey. In G. Rinaldi M. Jünger, G. Reinelt, editor, *Combinatorial optimization - eureka, you shrink!*, pages 185–207. Springer-Verlag, New York, 2003.

J. Yi and A. Scheller-Wolf. Vehicle routing with time windows and time-dependent rewards: A problem of american red cross. *Manufacturing and Operations Science Management*, 5:73–77, 2003.

VITA

EDA YÜCEL was born in Denizli, on January, 1982. She graduated with a B.S. degree from Bilkent University Department of Computer Engineering in 2003, with a 3rd rank. Up to July 2004, she worked in Milsoft Software Technologies as a design and software engineer. She started the M.Sc. program in Industrial Engineering at Koç University in 2004, and she received her M.Sc. degree in 2006. Between 2007 and 2009, she worked as an intercorporate consultant for material requirements planning in Temsa Global, which is a leading manufacturer in Turkish automotive industry. Meanwhile, she joined the Ph.D. program in Industrial Engineering and Operations Management at Koç University. During her Ph.D. studies, she also worked as an instructor in the Department of Computer Engineering in Çukurova University, where she taught courses on theory of computation, algorithms, and computational complexity.

Her research interests are in mathematical programming and optimization, especially in the areas of scheduling and transportation logistics. She is also interested in algorithm design and analysis.

She is married to Ahmet Oğuz and they have two sons, Mert Selim and Mete Kerem.